Parallel Spooky Pebbling Makes Regev Factoring More Practical

[ePrint:2025/1887]

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

[1]MIT

[2]Harvard

May 11, 2026

Quantum integer factorization

Shor ’94: Factoring n-bit integers in poly(n) time on a quantum computer

Quantum integer factorization

Shor ’94: Factoring n-bit integers in poly(n) time on a quantum computer

- What is the precise complexity of quantum factoring?

Quantum integer factorization

Shor '94: Factoring n-bit integers in poly(n) time on a quantum computer

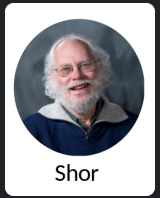
- What is the precise complexity of quantum factoring?
- What are the time/space tradeoffs?

Quantum integer factorization

Shor '94: Factoring n-bit integers in poly(n) time on a quantum computer

- What is the precise complexity of quantum factoring?
- What are the time/space tradeoffs?
- **What does it take in practice?**

Circuits for factoring general-form numbers

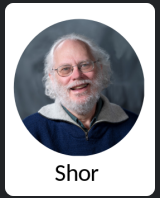


Core of Shor's algorithm

To factor an n-bit number N, run quantum period finding on

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

Circuits for factoring general-form numbers



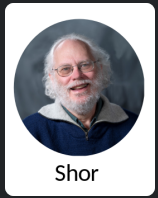
Core of Shor's algorithm

To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$

Circuits for factoring general-form numbers



Circuit	Gates	Qubits
Shor '95	$\widetilde{O}(n^2)$	$\widetilde{O}(n)$

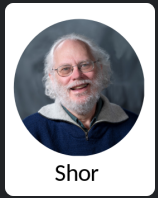
Core of Shor's algorithm

To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$

Circuits for factoring general-form numbers


Shor

Core of Shor's algorithm

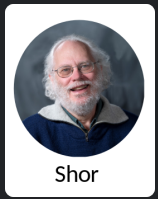
To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$

Circuit	Gates	Qubits
Shor '95	$\widetilde{O}(n^2)$	$\widetilde{O}(n)$
Beauregard '03	$O(n^3)$	$2n + O(1)$

Circuits for factoring general-form numbers


Shor

Core of Shor's algorithm

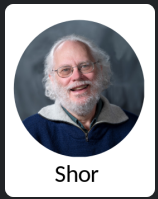
To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$

Circuit	Gates	Qubits
Shor '95	$\widetilde{O}(n^2)$	$\widetilde{O}(n)$
Beauregard '03	$O(n^3)$	$2n + O(1)$
Meyer et al. '24	$O(n^{2+\varepsilon})$	$2n + o(n)$

Circuits for factoring general-form numbers


Shor

Core of Shor's algorithm

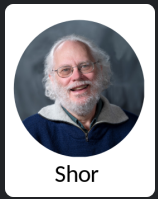
To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$

Circuit	Gates	Qubits
Shor '95	$\widetilde{O}(n^2)$	$\widetilde{O}(n)$
Beauregard '03	$O(n^3)$	$2n + O(1)$
Meyer et al. '24	$O(n^{2+\varepsilon})$	$2n + o(n)$
Chevignard et al. '24	$\widetilde{O}(n^3)$	$n/2 + o(n)$

Circuits for factoring general-form numbers


Shor

Core of Shor's algorithm

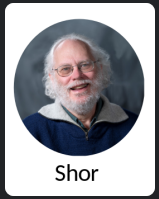
To factor an n-bit number N, run quantum period finding on

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

Bottleneck: $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$

Circuit	Gates	Qubits
Shor '95	$\widetilde{O}(n^2)$	$\widetilde{O}(n)$
Beauregard '03	$O(n^3)$	$2n + O(1)$
Meyer et al. '24	$O(n^{2+\varepsilon})$	$2n + o(n)$
Chevignard et al. '24	$\widetilde{O}(n^3)$	$n/2 + o(n)$
↳ Gidney '25	[concrete improvements]	

Circuits for factoring general-form n-bit numbers



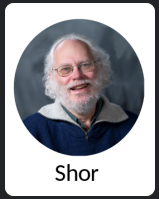
Shor

Core of Shor's algorithm

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

$$\log x \sim O(n)$$

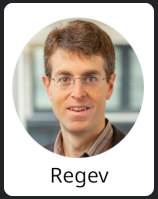
Circuits for factoring general-form n-bit numbers



Core of Shor's algorithm

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

$\log x \sim O(n)$



Core of Regev's algorithm

$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$

$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$

Roadmap

Previous approaches to Regev

Pebble games

Our work: parallel spooky pebble games

Roadmap

Previous approaches to Regev

Pebble games

Our work: parallel spooky pebble games

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \pmod{N}$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \qquad (\bmod N)$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \xrightarrow{\text{square}} a^4 \qquad (\text{mod } N)$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \xrightarrow{\text{square}} a^4 \xrightarrow{\text{mult.}} a^5 \qquad (\text{mod } N)$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \xrightarrow{\text{square}} a^4 \xrightarrow{\text{mult.}} a^5 \xrightarrow{\text{square}} a^{10} \qquad (\text{mod } N)$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \xrightarrow{\text{square}} a^4 \xrightarrow{\text{mult.}} a^5 \xrightarrow{\text{square}} a^{10} \xrightarrow{\text{mult.}} a^{11} \pmod{N}$$

Cost of computing $a^x \bmod N$

Example: how to compute $a^{11} \bmod N$

Repeated squaring:

$$a \xrightarrow{\text{square}} a^2 \xrightarrow{\text{square}} a^4 \xrightarrow{\text{mult.}} a^5 \xrightarrow{\text{square}} a^{10} \xrightarrow{\text{mult.}} a^{11} \pmod{N}$$

Cost: $O(\log x)$ multiplications of n-bit integers

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$$\log x \sim O(n)$$

Regev

$$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$$

$$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$$

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$\log x \sim O(n)$

Cost: $O(n)$ mults.

Regev

$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$

$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$$\log x \sim O(n)$$

Cost: $O(n)$ mults.

Regev

$$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$$

$$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$$

→ Each $a_i^{x_i}$ costs $O\left(\sqrt{n}\right)$ mults

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$$\log x \sim O(n)$$

Cost: $O(n)$ mults.

Regev

$$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$$

$$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$$

→ Each $a_i^{x_i}$ costs $O\left(\sqrt{n}\right)$ mults

→ Regev magic ⇒ pick small a_i and get...

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$$\log x \sim O(n)$$

Cost: $O(n)$ mults.

Regev

$$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$$

$$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$$

→ Each $a_i^{x_i}$ costs $O\left(\sqrt{n}\right)$ mults

→ Regev magic ⇒ pick small a_i and get...

Cost: $O\left(\sqrt{n}\right)$ mults!

Regev's gate savings

→ Computing $a^x \bmod N$ costs $O(\log x)$ multiplications

Shor

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

$$\log x \sim O(n)$$

Cost: $O(n)$ mults.

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

Regev

$$f(\vec{x}) = \prod_{i=1}^{d} a_i^{x_i} \bmod N$$

$$d \sim O\left(\sqrt{n}\right) \qquad \log x \sim O\left(\sqrt{n}\right)$$

→ Each $a_i^{x_i}$ costs $O\left(\sqrt{n}\right)$ mults

→ Regev magic ⇒ pick small a_i and get...

Cost: $O\left(\sqrt{n}\right)$ mults!

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

Making repeated squaring quantum

Repeated squaring:

$$a \to a^2 \to a^4 \to a^5 \to a^{10} \to a^{11} \pmod{N}$$

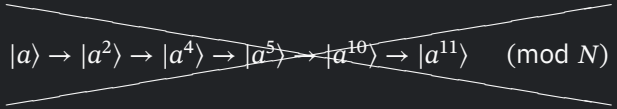
Making repeated squaring quantum

Quantum repeated squaring?

$$|a\rangle \to |a^2\rangle \to |a^4\rangle \to |a^5\rangle \to |a^{10}\rangle \to |a^{11}\rangle \quad (\text{mod } N)$$

Making repeated squaring quantum

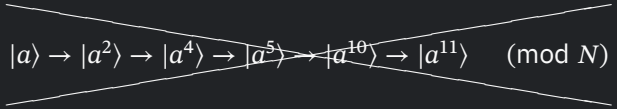
Quantum repeated squaring?



Issue: in-place squaring mod$N$ is not reversible!

Making repeated squaring quantum

Quantum repeated squaring?



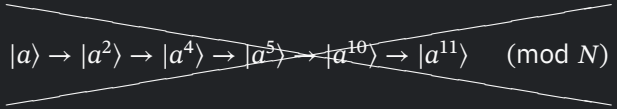
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



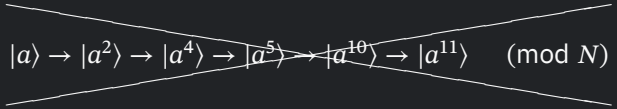
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



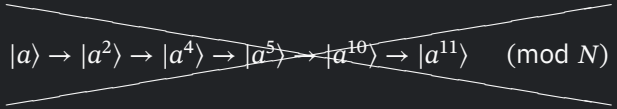
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



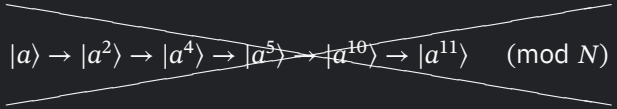
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|0\rangle\,|0\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



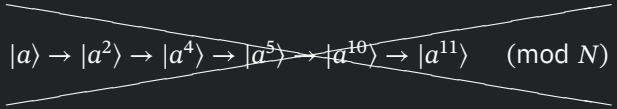
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|a^5\rangle\,|0\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



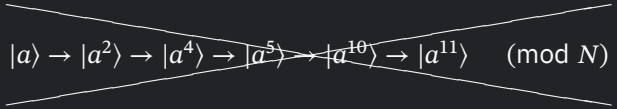
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|a^5\rangle\,|a^{10}\rangle\,|0\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



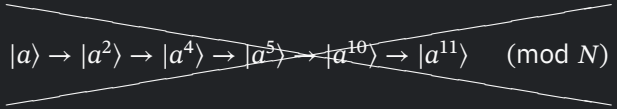
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|a^5\rangle\,|a^{10}\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



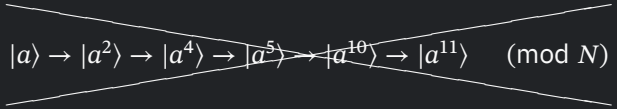
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|a^5\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



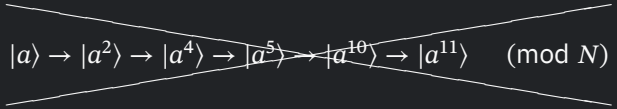
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|a^4\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



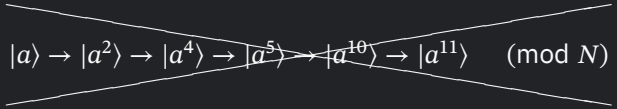
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|a^2\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



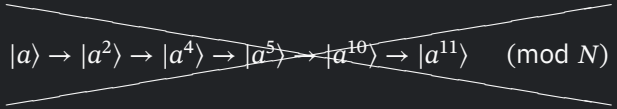
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|a\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



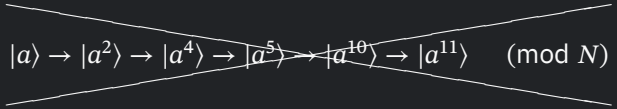
Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

Making repeated squaring quantum

Quantum repeated squaring?



Issue: in-place squaring mod$N$ is not reversible!

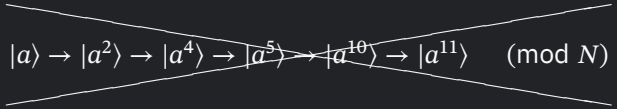
One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

✅ Reversible

Making repeated squaring quantum

Quantum repeated squaring?



Issue: in-place squaring mod$N$ is not reversible!

One could **temporarily store all intermediate values** (used in Regev's original paper):

$$|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|a^{11}\rangle \pmod N$$

✅ Reversible 😩 *Huge* qubit count

Irreversible sequential algorithms

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

Irreversible sequential algorithms

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

Option 1

Find an algorithm for f with *reversible steps*

Option 2

Accept irreversibility

Irreversible sequential algorithms

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

Option 1

Find an algorithm for f with *reversible steps*

Shor:
Steps are ✅ *in-place* and ✅ *cheap*

Option 2

Accept irreversibility

Irreversible sequential algorithms

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

Option 1

Find an algorithm for f with *reversible steps*

Shor:
Steps are ✅ *in-place* and ✅ *cheap*

Option 2

Accept irreversibility

Regev:
Steps are ❌ *out-of-place* but ✅ *cheap*

Irreversible sequential algorithms

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

Option 1

Find an algorithm for f with *reversible steps*

Shor:
Steps are ✅ *in-place* and ✅ *cheap*

Regev + RV24:
Steps are ✅ *in-place* but ❌ *expensive*

Option 2

Accept irreversibility

Regev:
Steps are ❌ *out-of-place* but ✅ *cheap*

Irreversible sequential algorithms

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

Option 1

Find an algorithm for f with *reversible steps*

Shor:
Steps are ✅ *in-place* and ✅ *cheap*

Regev + RV24:
Steps are ✅ *in-place* but ❌ *expensive*

Option 2

Accept irreversibility

Regev:
Steps are ❌ *out-of-place* but ✅ *cheap*

This talk:
Better manage Regev's intermediate values!

Roadmap

Previous approaches to Regev

Pebble games

Our work: parallel spooky pebble games

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

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

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

$x \quad x_1 \quad x_2 \quad x_3 \quad f(x)$

State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

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

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

State: $|x\rangle\,|x_1\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

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

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

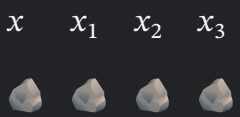
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

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

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



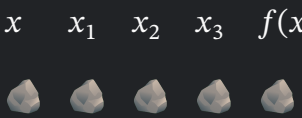
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

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

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



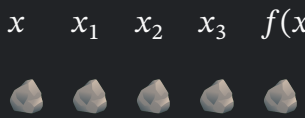
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|f(x)\rangle$

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*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



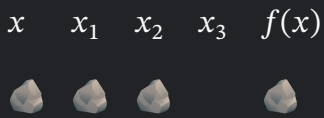
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|f(x)\rangle$

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*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



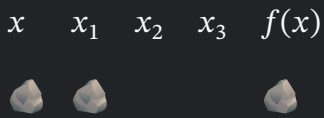
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|0\rangle\,|f(x)\rangle$

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*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



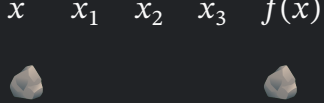
State: $|x\rangle\,|x_1\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

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*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

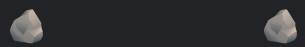
[Bennett ’89] introduced pebble games:

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

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

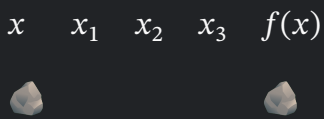
[Bennett '89] introduced pebble games:

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

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x)) \cdots)$
and $x_i = f_i(\cdots)$



Naive strategy

Time cost (# steps): $2k - 1$ (optimal) 😃

Pebble games for reversible computation

[Bennett '89] introduced pebble games:

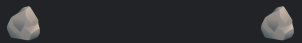
- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x)) \cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$



Naive strategy

Time cost (# steps): $2k - 1$ (optimal) 😃
Space (# pebbles): $O(k)$ 😭

Pebble games for reversible computation

[Bennett '89] introduced pebble games:

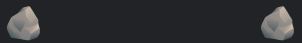
- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$



Naive strategy

Time cost (# steps): $2k - 1$ (optimal) 😃
Space (# pebbles): $O(k)$ 😭

→ Regev: $k = O(\sqrt{n})$ and n qubits per pebble ⇒ $O(n^{3/2})$ qubits

Using less space: a recursive strategy

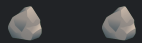
1. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

Using less space: a recursive strategy

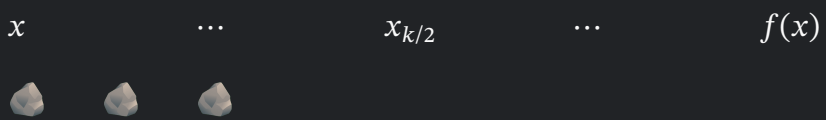
1. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$



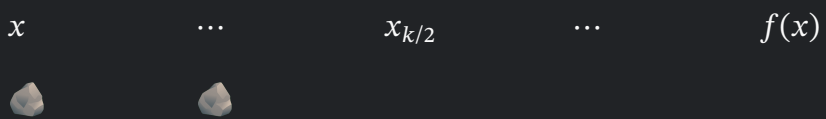
Using less space: a recursive strategy

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



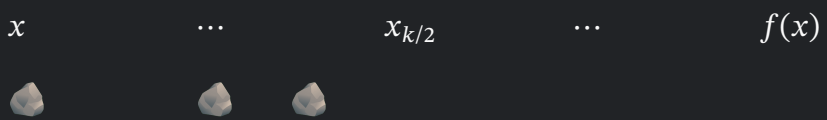
Using less space: a recursive strategy

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



Using less space: a recursive strategy

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



Using less space: a recursive strategy

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

Using less space: a recursive strategy

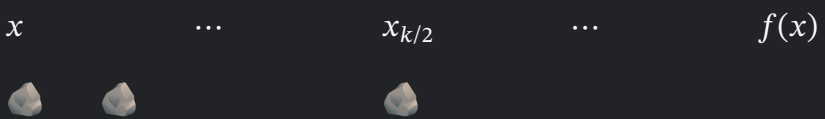
1. Pebble game from start to $k/2$

Using less space: a recursive strategy

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

Using less space: a recursive strategy

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



Using less space: a recursive strategy

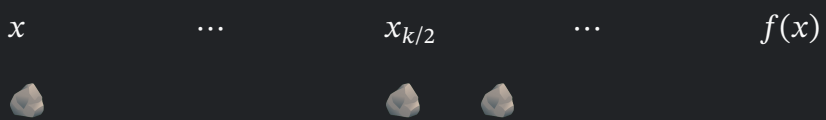
1. Pebble game from start to $k/2$

Using less space: a recursive strategy

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

Using less space: a recursive strategy

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

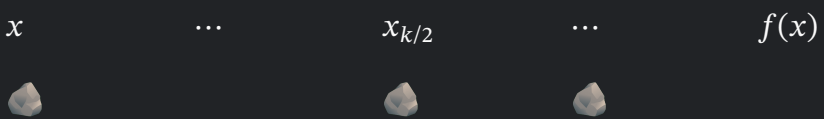


Using less space: a recursive strategy

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

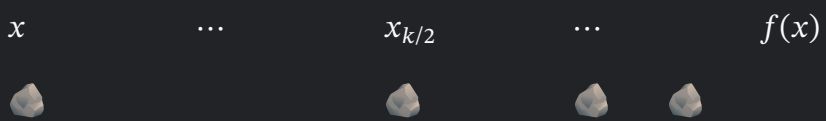
Using less space: a recursive strategy

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



Using less space: a recursive strategy

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

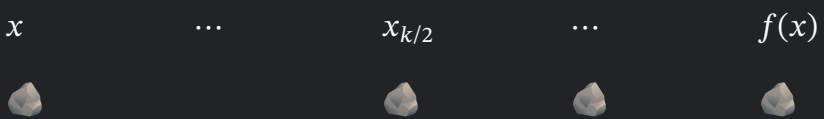


Using less space: a recursive strategy

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

Using less space: a recursive strategy

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



Using less space: a recursive strategy

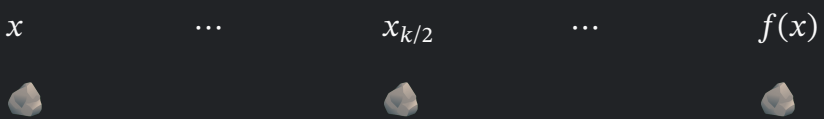
1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

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

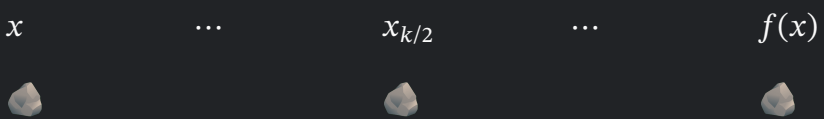
Using less space: a recursive strategy

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



Using less space: a recursive strategy

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

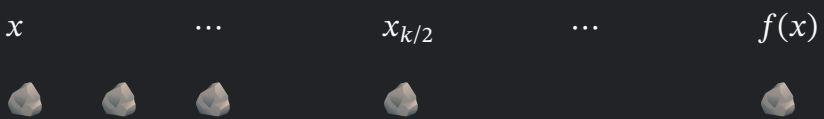


Using less space: a recursive strategy

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

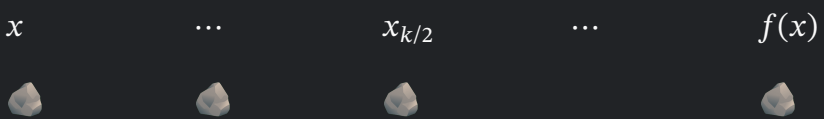
Using less space: a recursive strategy

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



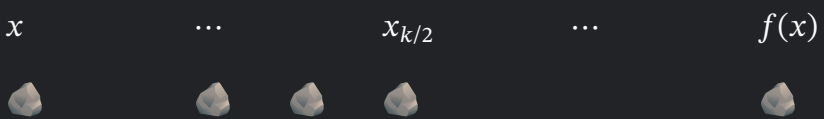
Using less space: a recursive strategy

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



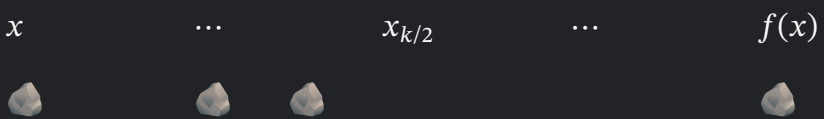
Using less space: a recursive strategy

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



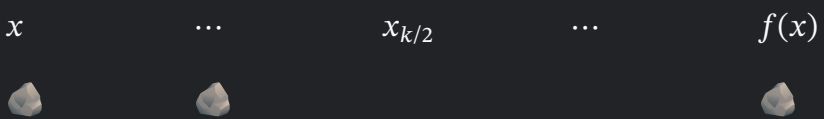
Using less space: a recursive strategy

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



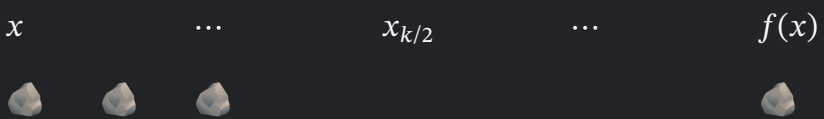
Using less space: a recursive strategy

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



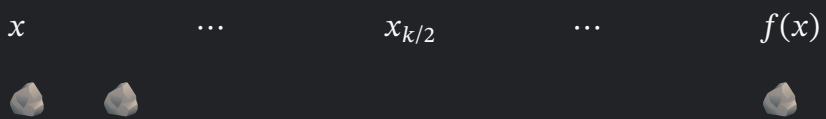
Using less space: a recursive strategy

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



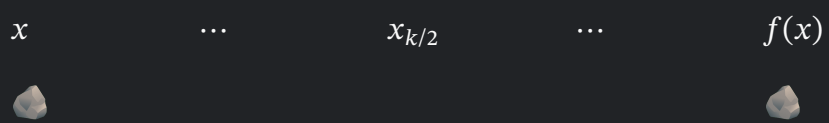
Using less space: a recursive strategy

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



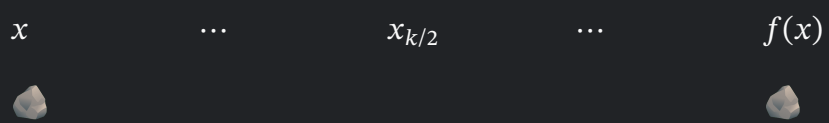
Using less space: a recursive strategy

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



Using less space: a recursive strategy

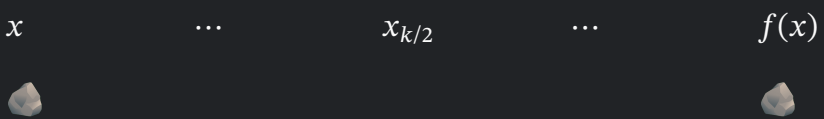
1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



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

Using less space: a recursive strategy

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

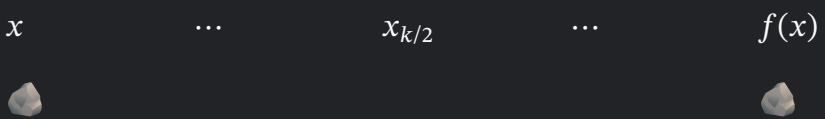


Space usage (# pebbles): $O(\log k)$ 😃

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

Using less space: a recursive strategy

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

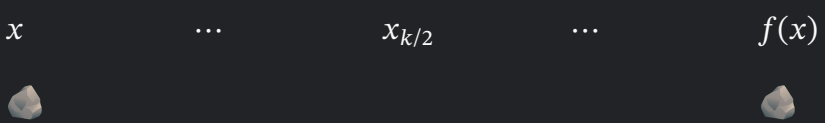


Space usage (# pebbles): $O(\log k)$ 😃

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

Using less space: a recursive strategy

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



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

Time cost (# steps): $O(k^{\log_2 3}) \approx O(k^{1.58\cdots})$ steps 😢
Generalization: space $O(2^{1/\epsilon} \log k)$, time $O(k^{1+\epsilon})$ 😐

Changing the rules of the game

Resources at our disposal:

Changing the rules of the game

Resources at our disposal:

1. Parallelism

Changing the rules of the game

Resources at our disposal:

1. Parallelism
2. The spirit realm

Changing the rules of the game

Resources at our disposal:

1. Parallelism
2. ~~The spirit realm~~ Quantum!

Changing the rules of the game

Resources at our disposal:

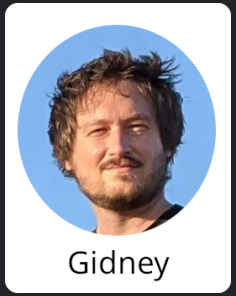
1. Parallelism
2. ~~The spirit realm~~ Quantum!

Pebble games using these resources *independently* have been explored—
what if we combine them?

Spookiness: a quantum resource

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

Spookiness: a quantum resource



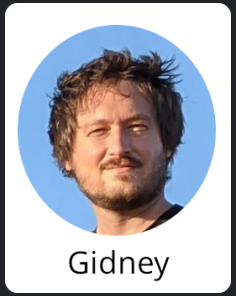
"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

Spookiness: a quantum resource



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

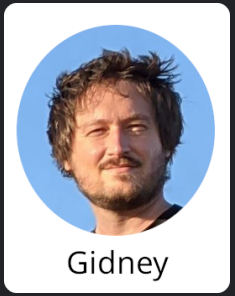
Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

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

Spookiness: a quantum resource



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

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

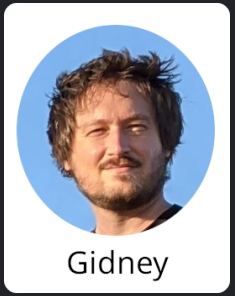
what if we apply $H^{\otimes n}$ and then measure it?

Get phase

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

where d is *classically-known*.

Spookiness: a quantum resource



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

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

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

Get phase

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

where d is *classically-known*.

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

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

Rules:

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

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

Rules:

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

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

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

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

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
- exorcism is free when x_i has a pebble

The power of spooky pebbling

Recursive strategy

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

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. Spooky pebble game from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

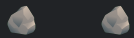
x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

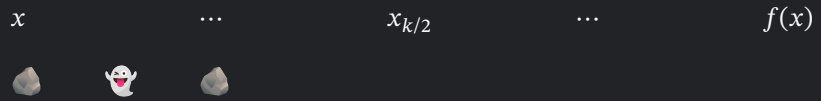
x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$



The power of spooky pebbling

Spooky recursive strategy

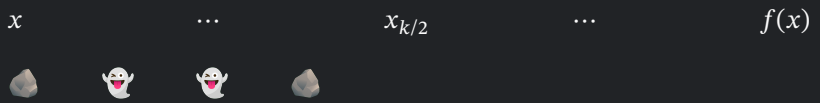
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

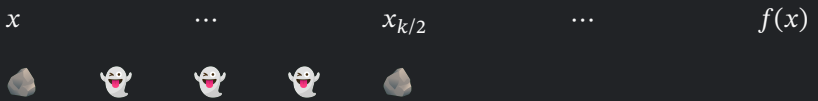
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

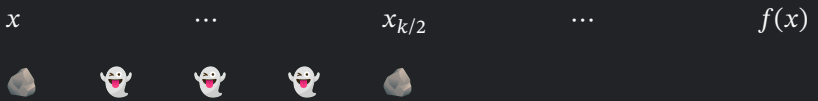
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

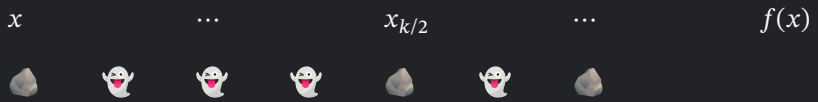
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

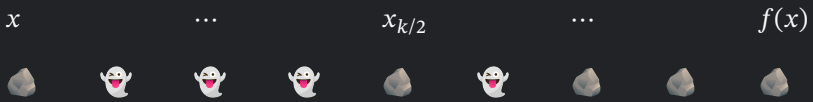
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

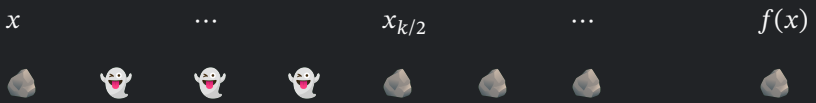
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

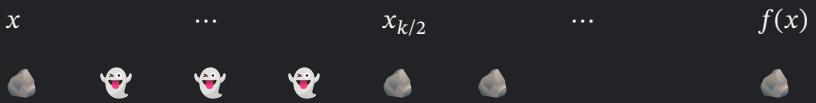
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

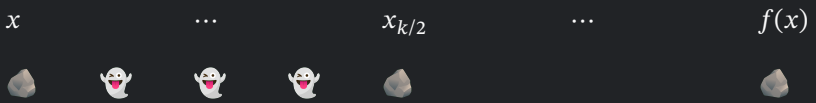
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

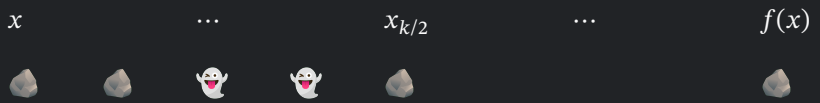
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

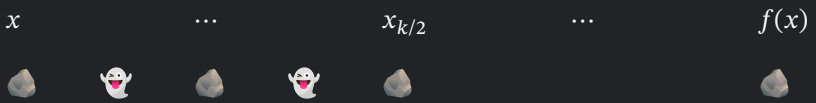
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

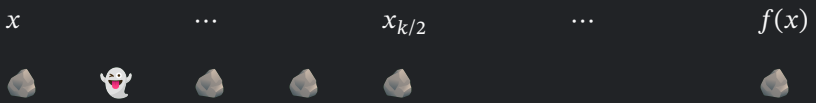
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

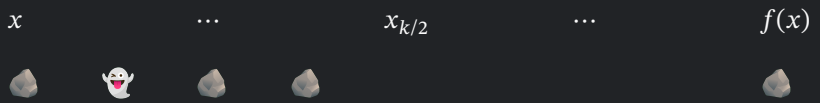
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

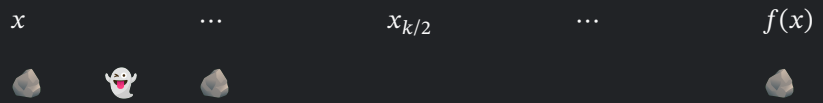
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

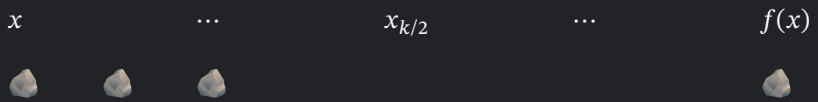
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

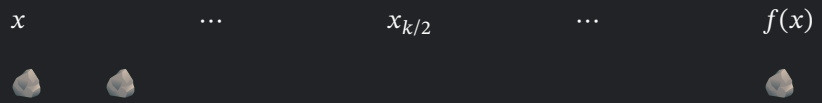
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

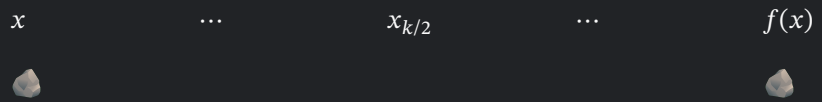
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

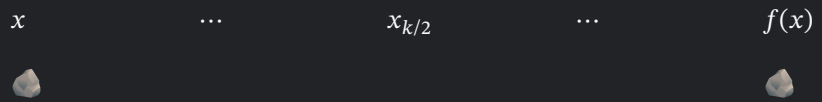
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

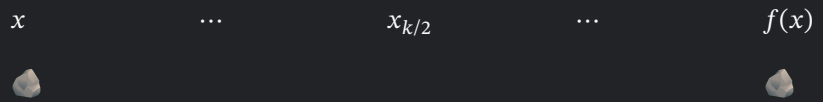


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps

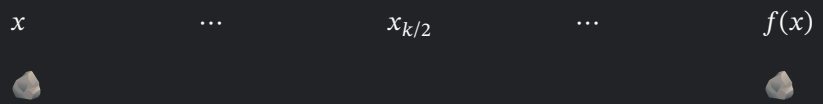


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps

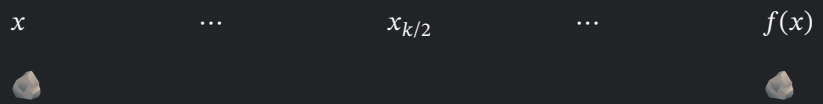


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps



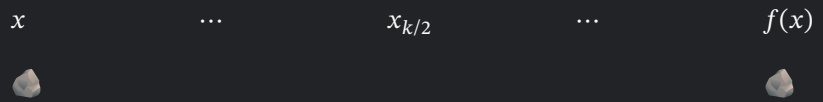
Space: $O(\log k)$ pebbles 😃

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

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps



Space: $O(\log k)$ pebbles 😃

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

The power of spooky pebbling

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett ’89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
BHL ’22	Y	N	$O(4^{\sqrt{\log k}})$	$O(k)$
Gidney ’19 KSS ’25	N	Y	$O(\log k)$	$O(k \log k)$

The power of spooky pebbling

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett ’89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
BHL ’22	Y	N	$O(4^{\sqrt{\log k}})$	$O(k)$
Gidney ’19 KSS ’25	N	Y	$O(\log k)$	$O(k \log k)$

Is the best of both worlds possible?

Roadmap

Previous approaches to Regev

Pebble games

Our work: parallel spooky pebble games

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 0

Max. pebble count: 1

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 1

Max. pebble count: 2

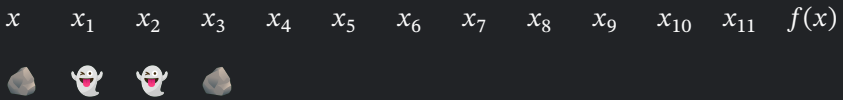
Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 2

Max. pebble count: 2

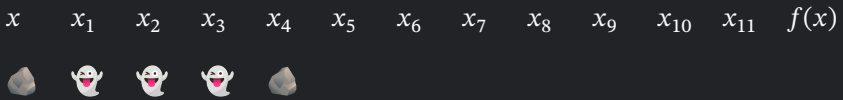
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 3

Max. pebble count: 2

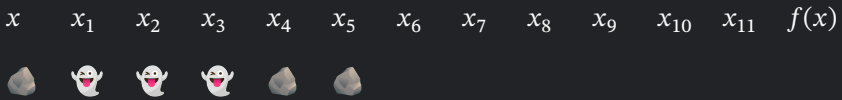
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 4

Max. pebble count: 2

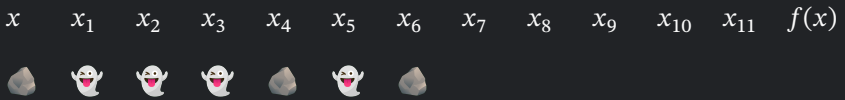
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 5

Max. pebble count: 3

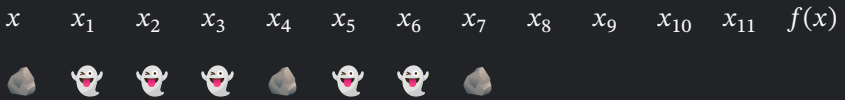
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 6

Max. pebble count: 3

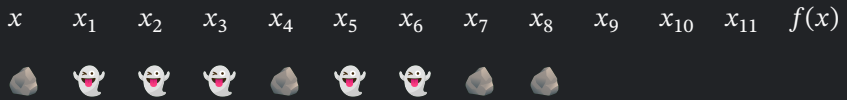
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 7

Max. pebble count: 3

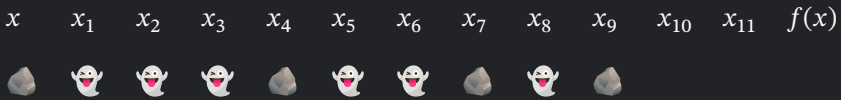
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 8

Max. pebble count: 4

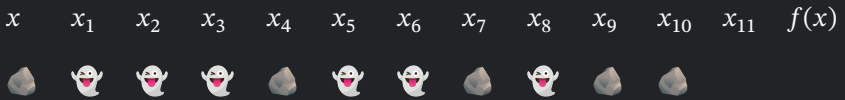
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 9

Max. pebble count: 4

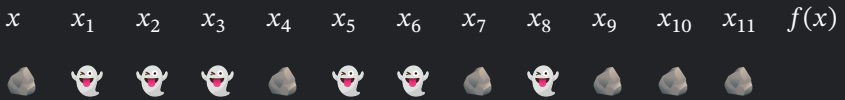
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 10

Max. pebble count: 5

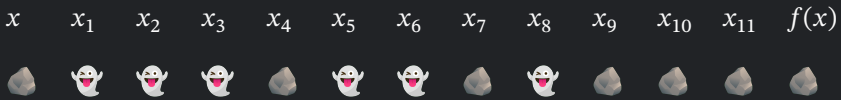
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 11

Max. pebble count: 6

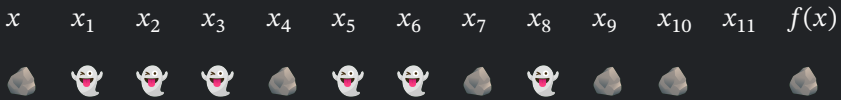
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 12

Max. pebble count: 7

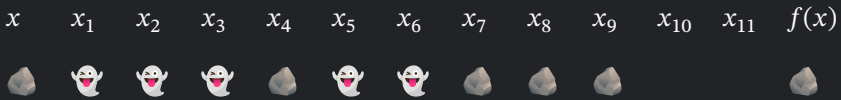
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 13

Max. pebble count: 7

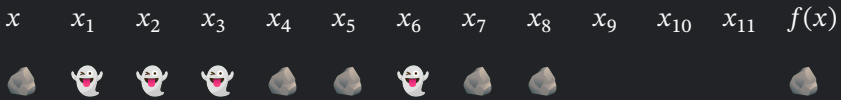
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 14

Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 15

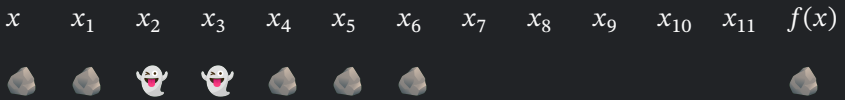
Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$

Step number: 16

Max. pebble count: 7

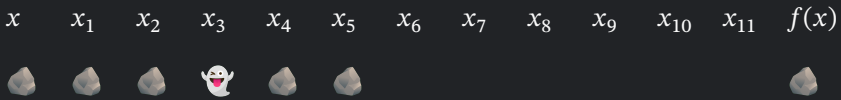
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 17

Max. pebble count: 7

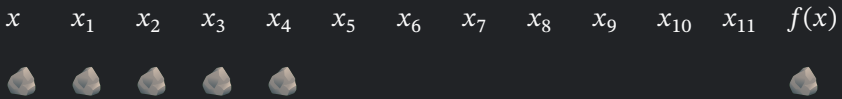
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 18

Max. pebble count: 7

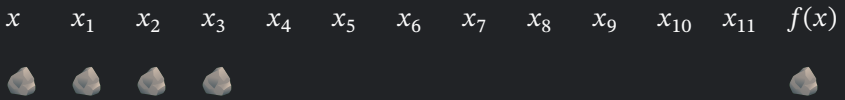
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 19

Max. pebble count: 7

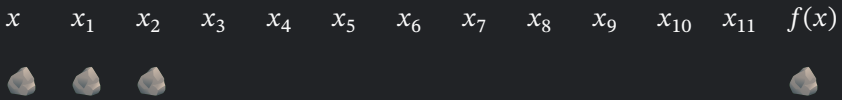
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 20

Max. pebble count: 7

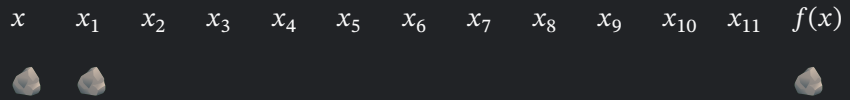
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 21

Max. pebble count: 7

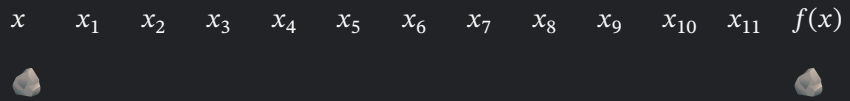
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 22

Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 23

Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

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

Max. pebble count: 7

Our results

Explicit algorithms for pebbling to length k:

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett '89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
BHL '22	Y	N	$O(4^{\sqrt{\log k}})$	$O(k)$
Gidney '19 KSS '25	N	Y	$O(\log k)$	$O(k \log k)$
Our work	**Y**	**Y**	$\approx 2.47 \log k$	$2k-1$

Automated search

What if we have too few pebbles for depth $2k - 1$?

Automated search

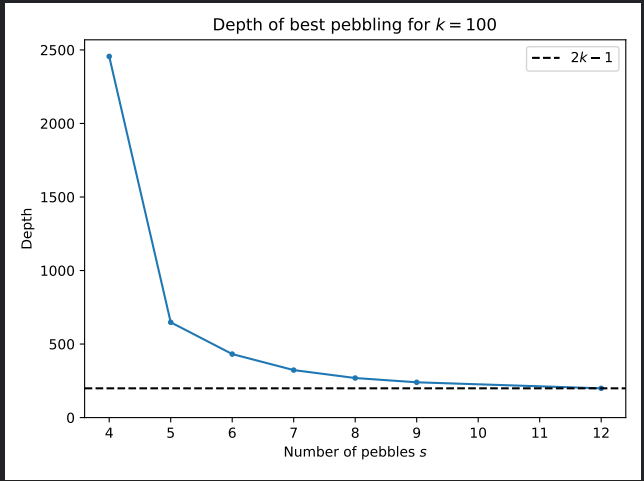
What if we have too few pebbles for depth $2k - 1$?

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

Automated search

What if we have too few pebbles for depth $2k - 1$?

- Highly optimized **A* search** written in Julia
- On input number of pebbles s and length k, finds absolutely optimal solution



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
Regev + RV24	700	1400	$\geq 13n$

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
Regev + RV24	700	1400	$\geq 13n$
Standard Shor (e.g. Gidney + Ekerå ’19)	444	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
Regev + RV24	700	1400	$\geq 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	444	$2n - 3n$
Standard Shor + some parallelism	186	896	$\approx 14n$

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
Regev + RV24	700	1400	$\geq 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	444	$2n - 3n$
Standard Shor + some parallelism	186	896	$\approx 14n$
Our results (more pebbles)	200	381	$15n$

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
Regev + RV24	700	1400	$\geq 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	444	$2n - 3n$
Standard Shor + some parallelism	186	896	$\approx 14n$
Our results (more pebbles)	200	381	$15n$
Our results (fewer pebbles)	561	600	$8n$

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
Regev + RV24	700	1400	$\geq 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	444	$2n - 3n$
Standard Shor + some parallelism	186	896	$\approx 14n$
Our results (more pebbles)	200	381	$15n$
Our results (fewer pebbles)	561	600	$8n$

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

Takeaways

- Concretely best Regev variant to date

Takeaways

- Concretely best Regev variant to date
- Shor still somewhat more efficient

Takeaways

- Concretely best Regev variant to date
- Shor still somewhat more efficient
- Also, qubit count seems to be most important *near-term*

Takeaways

- Concretely best Regev variant to date
- Shor still somewhat more efficient
- Also, qubit count seems to be most important *near-term*
- “This paper creates **slack**”—for further optimization!

Takeaways

- Concretely best Regev variant to date
- Shor still somewhat more efficient
- Also, qubit count seems to be most important *near-term*
- “This paper creates **slack**”—for further optimization!
- Efficiently evaluating sequential algorithms is **widely applicable** beyond factoring

Takeaways

- Concretely best Regev variant to date
- Shor still somewhat more efficient
- Also, qubit count seems to be most important *near-term*
- “This paper creates **slack**”—for further optimization!
- Efficiently evaluating sequential algorithms is **widely applicable** beyond factoring
- **Generalization:** parallel spooky pebbling on arbitrary graphs

INTELLIGENCE COMMUNITY
POSTDOCTORAL RESEARCH
FELLOWSHIP PROGRAM

Jane Street

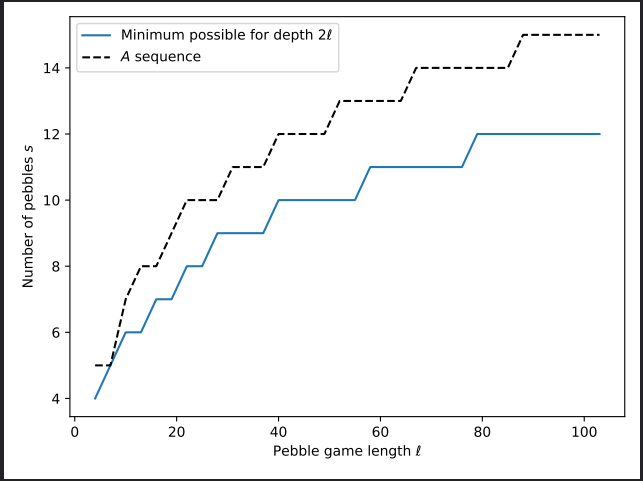
Thank you!

ePrint:2025/1887
arXiv:2510.08432

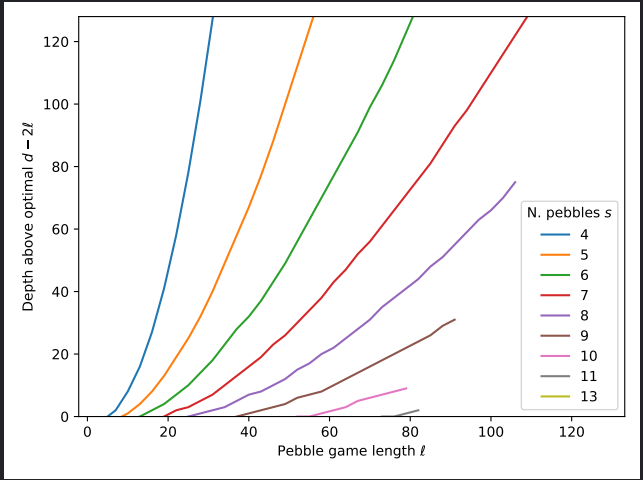
GDKM was supported by an appointment to the Intelligence Community Postdoctoral Research Fellowship Program at MIT administered by Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy and the Office of the Director of National Intelligence (ODNI).

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:

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

$|1\rangle$

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

Making Shor reversible

Break up x into its individual bits x_i:

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

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

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:

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

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

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

Making Shor reversible

Break up x into its individual bits x_i:

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

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

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

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

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