Analysis of the One Line Factoring Algorithm on Large Semiprimes

Tejas Gopalakrishna
Mentor: Yichi Zhang

May 18th, 2019
MIT PRIMES Conference

Introduction

What is a factoring algorithm?

Find a divisor of N.

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

Divide by primes $[2, 3, 5, 7]$

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

Divide by primes $[2, 3, 5, 7]$

$119/2$ not an integer.

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

Divide by primes $[2, 3, 5, 7]$

$119/2$ not an integer.

$119/3$ not an integer.

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

Divide by primes $[2, 3, 5, 7]$

$119/2$ not an integer.

$119/3$ not an integer.

$119/5$ not an integer.

The Naïve Algorithm – Trial Division

Factoring N

Divide by every prime in $[1 \ldots \sqrt{N}]$

Example: $N = 119$

Divide by primes $[2, 3, 5, 7]$

$119/2$ not an integer.

$119/3$ not an integer.

$119/5$ not an integer.

$119/7 = 17$ is an integer!

Simple Factoring Algorithm: Fermat

Factoring N

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 $(a := a + 1)$
  $b := a^2 - N$
- When b is a square, then
  $(a - \sqrt{b})$ is a factor.

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 ($a := a + 1$)
  $b := a^2 - N$
- When b is a square, then
  $(a - \sqrt{b})$ is a factor.

- $a := \lceil \sqrt{119} \rceil = 11$

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 ($a := a + 1$)
  $b := a^2 - N$
- When b is a square, then
  $(a - \sqrt{b})$ is a factor.

- $a := \lceil \sqrt{119} \rceil = 11$
- $b := 11^2 - 119 = 2$

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 ($a := a + 1$)
  $b := a^2 - N$
- When b is a square, then
  $(a - \sqrt{b})$ is a factor.

- $a := \lceil \sqrt{119} \rceil = 11$
- $b := 11^2 - 119 = 2$
- b (2) is not a square:
  $a := a + 1 = 12$
  $b := 12^2 - 119 = 25$

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 $(a := a + 1)$
  $b := a^2 - N$
- When b is a square, then
  $(a - \sqrt{b})$ is a factor.

- $a := \lceil \sqrt{119} \rceil = 11$
- $b := 11^2 - 119 = 2$
- b (2) is not a square:
  $a := a + 1 = 12$
  $b := 12^2 - 119 = 25$
- $b = 25$ is a square, so
  $12 - \sqrt{25} = 7$ is a factor.

Simple Factoring Algorithm: Fermat

Factoring N

- Let $a := \lceil \sqrt{N} \rceil$
- Let $b := a^2 - N$
- Repeat until b is a square:
  Increase a by 1 ($a := a + 1$)
  $b := a^2 - N$
- When b is a square, then $(a - \sqrt{b})$ is a factor.

- $a := \lceil \sqrt{119} \rceil = 11$
- $b := 11^2 - 119 = 2$
- b (2) is not a square:
  $a := a + 1 = 12$
  $b := 12^2 - 119 = 25$
- $b = 25$ is a square, so $12 - \sqrt{25} = 7$ is a factor.

Works because of square difference $x^2 - y^2 = (x + y)(x - y)$

One Line Factoring Algorithm?

One Line Factoring Algorithm?

Slower than the leading algorithms

One Line Factoring Algorithm?

Slower than the leading algorithms

Much less space required

“One Line” O.o

One line of PARI/GP…

```
OLF(x)=;i=1;while(i<x,if(issquare(ceil(sqrt(i*x)
)^2%x),return(gcd(x,floor(ceil(sqrt(i*x))-sqrt((
ceil(i*x))^2)%))))));i++)
```

The One Line Factoring Algorithm

Factoring N

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \% \ N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \ \% \ N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \% N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

- When $k = 1$, $m = 2$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \ \% \ N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

- When $k = 1$, $m = 2$
- When $k = 2$, $m = 18$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \% N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

- When $k = 1$, $m = 2$
- When $k = 2$, $m = 18$
- When $k = 3$, $m = 4$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \ \%\ N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

- When $k = 1$, $m = 2$
- When $k = 2$, $m = 18$
- When $k = 3$, $m = 4$
- Factor:
  $\mathrm{GCD}(119, \left\lceil \sqrt{119 \cdot 3} \right\rceil - \sqrt{4})$

The One Line Factoring Algorithm

Factoring N

Repeat for $k = 1$ to $k = N$:

- Let $m := \left\lceil \sqrt{N \cdot k} \right\rceil^2 \ \% \ N$
- If m is a square:
  Factor is
  $\mathrm{GCD}(N, \left\lceil \sqrt{N \cdot k} \right\rceil - \sqrt{m})$

Example: $N = 119$

- When $k = 1$, $m = 2$
- When $k = 2$, $m = 18$
- When $k = 3$, $m = 4$
- Factor:
  $\mathrm{GCD}(119, \left\lceil \sqrt{119 \cdot 3} \right\rceil - \sqrt{4})$
  $\mathrm{GCD}(119, 17) = 17$

Semiprimes

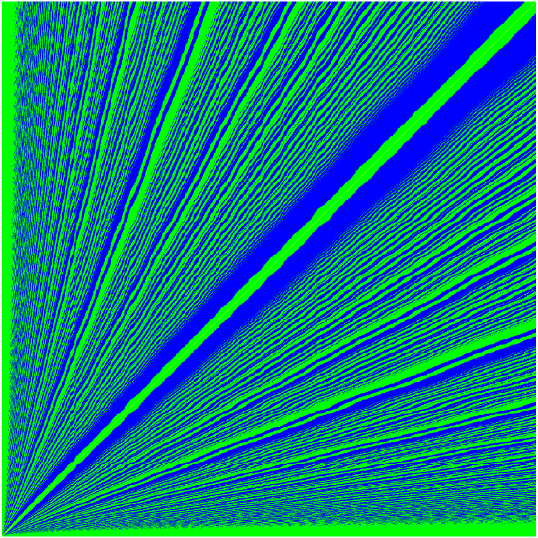
Factor numbers $N = pq$

Semiprimes

Factor numbers $N = pq$

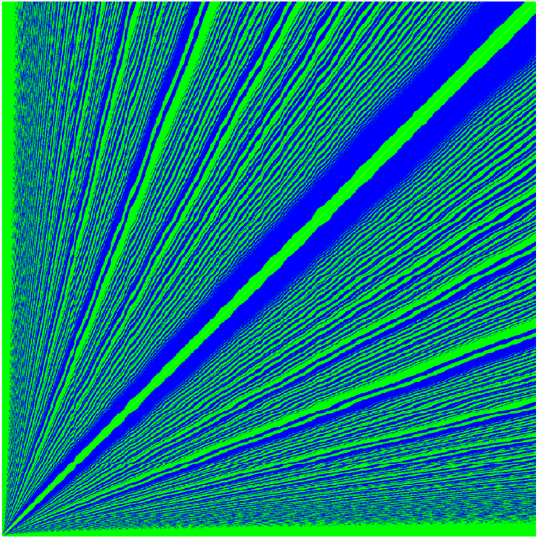
Applications in cryptography (like RSA)

Pretty Picture: Result of factoring pq



Pretty Picture: Result of factoring pq

Factoring pq:



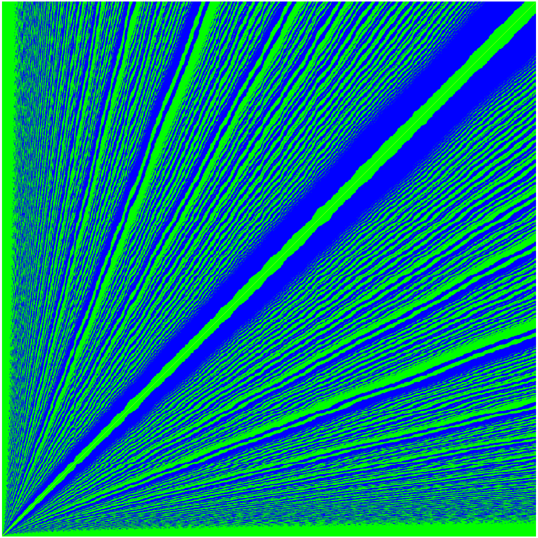
Pretty Picture: Result of factoring pq

Factoring pq:

- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes



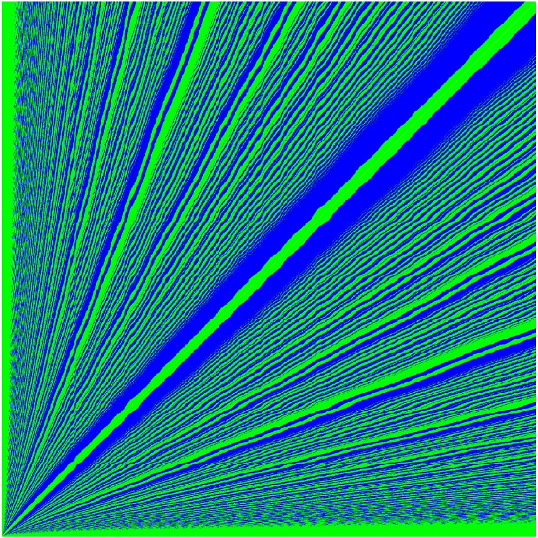
Pretty Picture: Result of factoring pq

Factoring pq:

- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes
- Green: Smaller prime returned



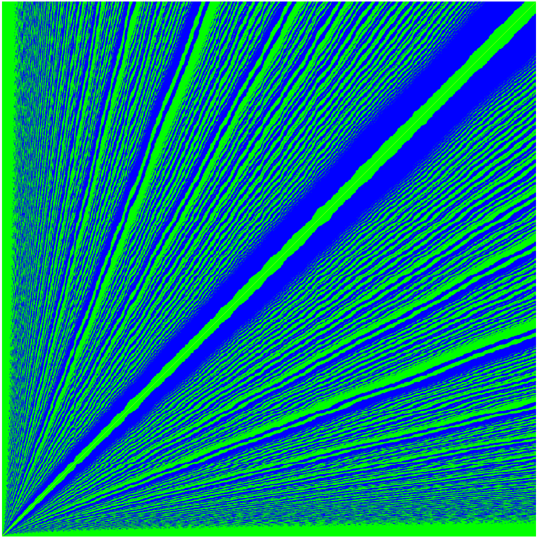
Pretty Picture: Result of factoring pq

Factoring pq:

- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes
- Green: Smaller prime returned
- If p,q are close: Smaller prime returned



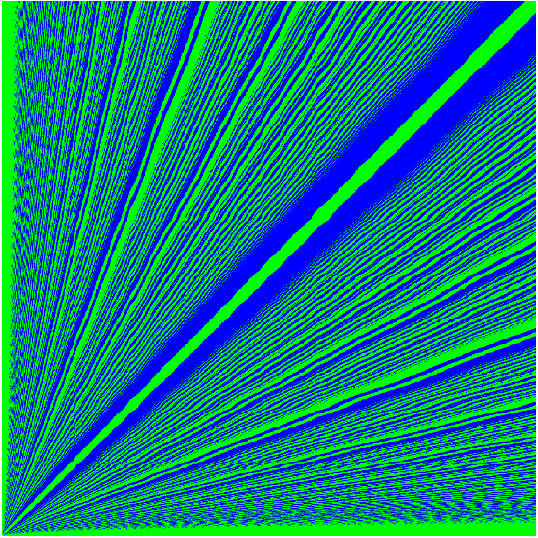
Pretty Picture: Result of factoring pq

Factoring pq:

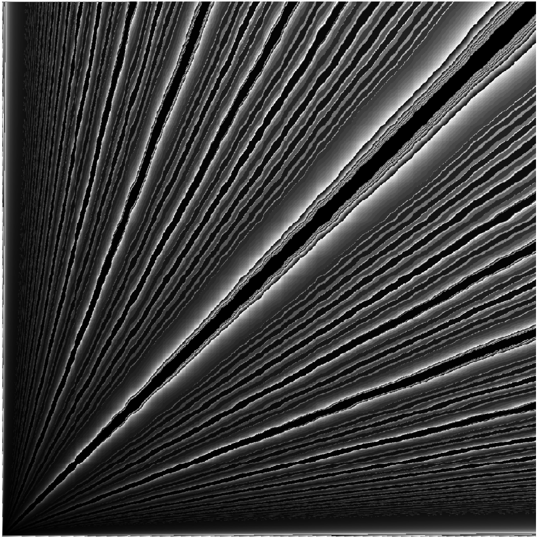
- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes
- Green: Smaller prime returned
- If p,q are close: Smaller prime returned
- Probability of green is $\sim 50\%$



Performance of OLF on semiprimes



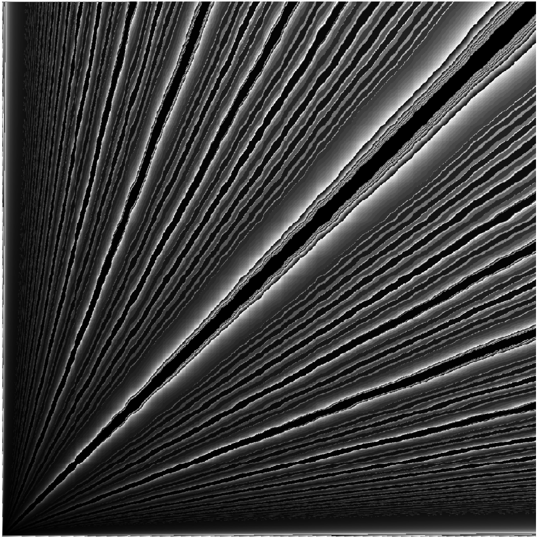
Performance of OLF on semiprimes

Number of iterations to factor pq:

- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes



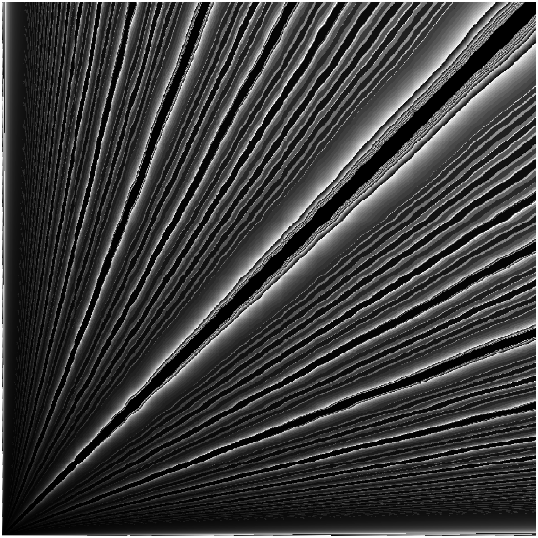
Performance of OLF on semiprimes

Number of iterations to factor pq:

- X-coordinate is prime p,

  Y-coordinate is prime q,

  p, q are first 1600 primes
- Points colored from black to white; Whiter means more iterations required



Improving Efficiency: Reduce Iterations?

The algorithm required trying **every number** from $k = 1$ to (at most) $k = N$

Improving Efficiency: Reduce Iterations?

The algorithm required trying **every number** from $k = 1$ to (at most) $k = N$

Can we skip some k?

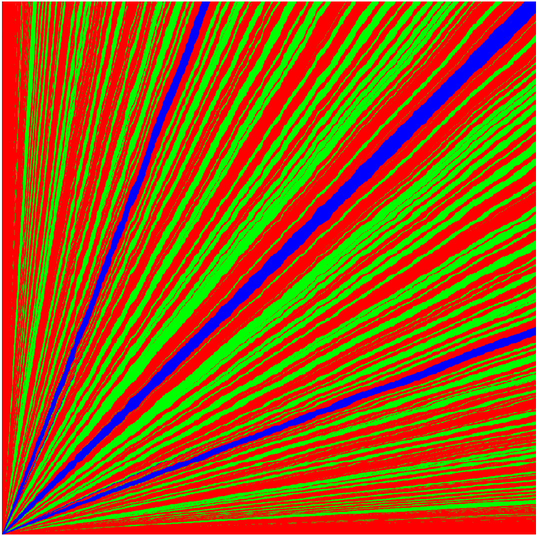
Improving Efficiency: Reduce Iterations?

The algorithm required trying **every number** from $k = 1$ to (at most) $k = N$

Can we skip some k?

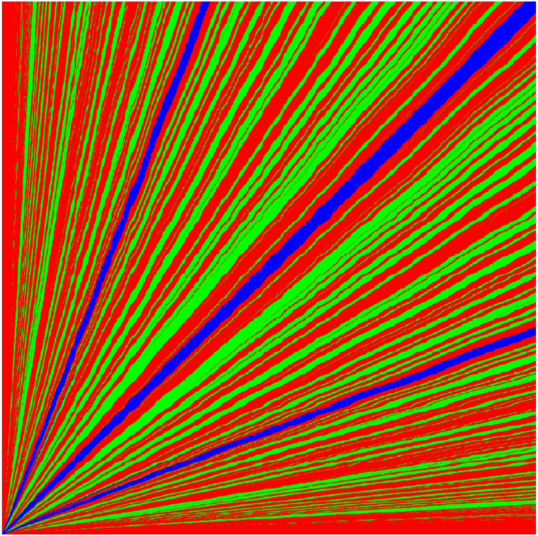
What if we just use squarefree k?

Pretty Picture: Where is squarefree OLF better?



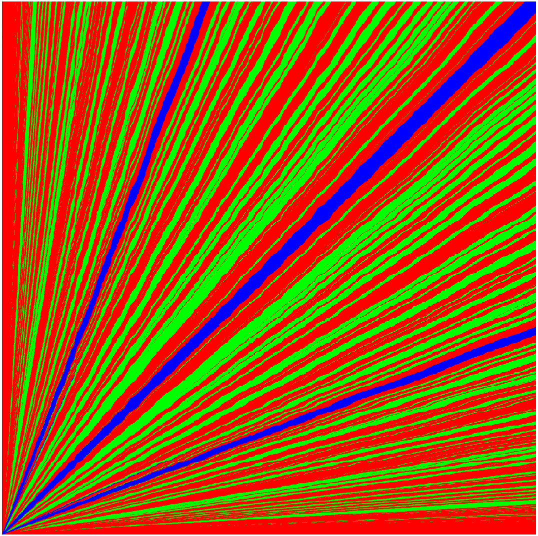
Pretty Picture: Where is squarefree OLF better?

- Green: Squarefree approach faster (fewer iterations)



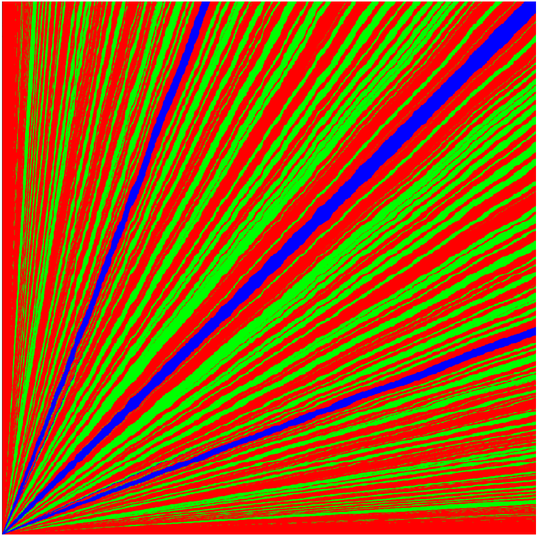
Pretty Picture: Where is squarefree OLF better?

- Green: Squarefree approach faster (fewer iterations)
- Distinct regions where this is more efficient

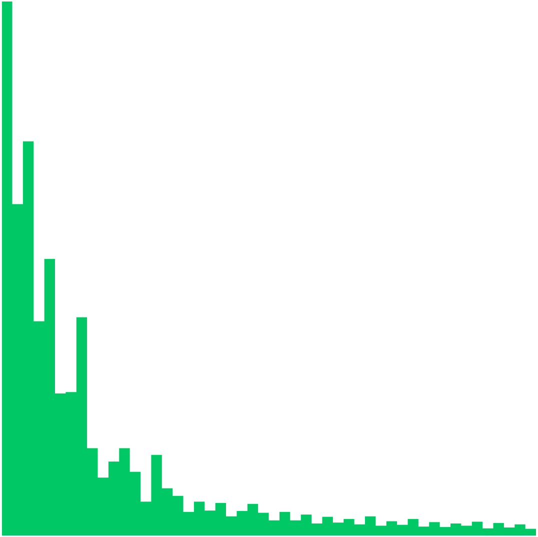


Pretty Picture: Where is squarefree OLF better?

- Green: Squarefree approach faster (fewer iterations)
- Distinct regions where this is more efficient
- Better on roughly $\sim 35.5\%$ of semiprimes

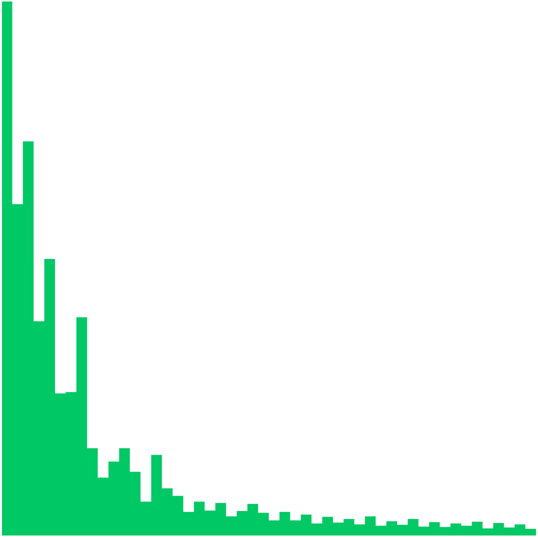


How many iterations to factor a general integer?



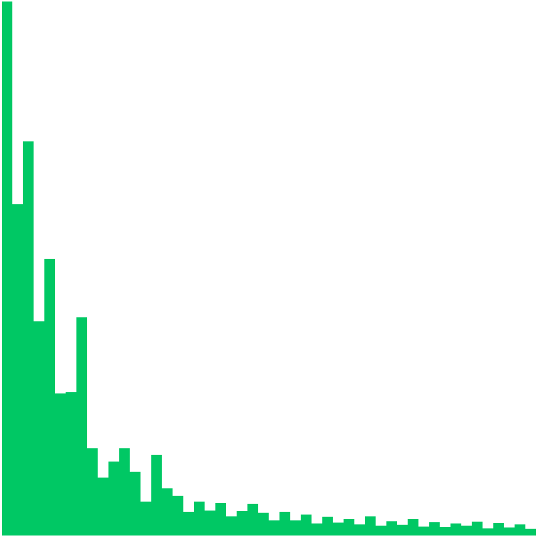
- k^{th} bar: Amount of integers that requires k iterations to factor

How many iterations to factor a general integer?



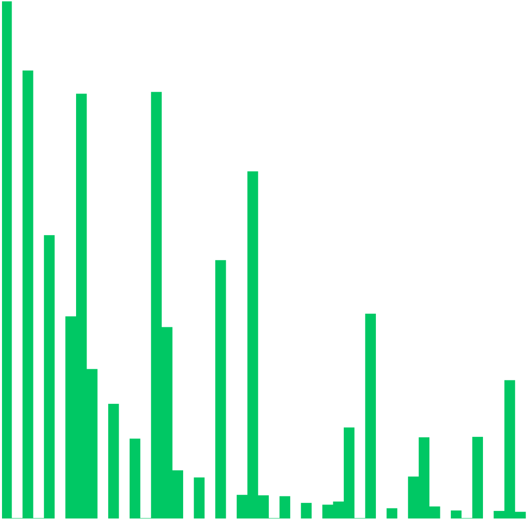
- k^{th} bar: Amount of integers that requires k iterations to factor
- Decreases rapidly

How many iterations to factor a general integer?



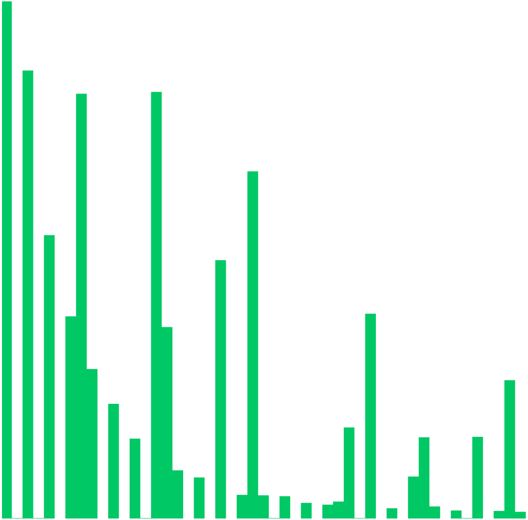
- k^{th} bar: Amount of integers that requires k iterations to factor
- Decreases rapidly
- Therefore, skipping k will not **always** help.

How many iterations to factor semiprimes?



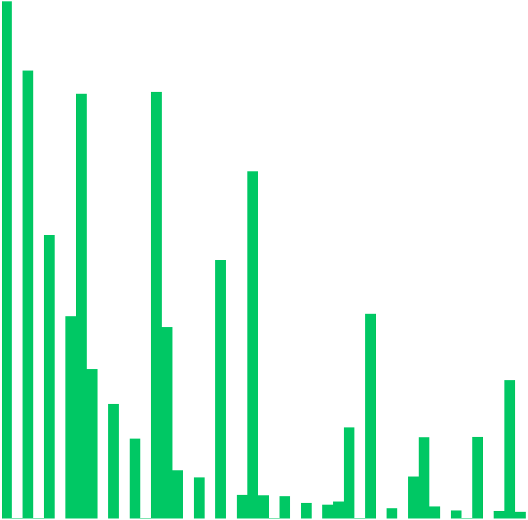
- However, the picture is different if only factoring semiprimes

How many iterations to factor semiprimes?



- However, the picture is different if only factoring semiprimes
- Many k not used.

How many iterations to factor semiprimes?



- However, the picture is different if only factoring semiprimes
- Many k not used.
- (Conjecture:) k only has to be $\{0, 1, 3, 5, 7\}$ modulo 8

Further Research

- What causes the strange bands?

Further Research

- What causes the strange bands?
- Can we precisely definine when the lower prime is returned?

Further Research

- What causes the strange bands?
- Can we precisely definine when the lower prime is returned?
- Prove the semiprime iterations conjecture.

Further Research

- What causes the strange bands?
- Can we precisely definine when the lower prime is returned?
- Prove the semiprime iterations conjecture.
- When can we skip k in the general algorithm (not just semiprimes)?

Further Research

- What causes the strange bands?
- Can we precisely definine when the lower prime is returned?
- Prove the semiprime iterations conjecture.
- When can we skip k in the general algorithm (not just semiprimes)?
- Anything else to make it faster!

Acknowledgements

Thanks a lot to...

- Mentor Yichi Zhang

Acknowledgements

Thanks a lot to...

- Mentor Yichi Zhang
- Dr. Stefan Wehmeir

Acknowledgements

Thanks a lot to...

- Mentor Yichi Zhang
- Dr. Stefan Wehmeir
- Dr. Tanya Khovanova

Acknowledgements

Thanks a lot to...

- Mentor Yichi Zhang
- Dr. Stefan Wehmeir
- Dr. Tanya Khovanova
- The PRIMES Program

Acknowledgements

Thanks a lot to...

- Mentor Yichi Zhang
- Dr. Stefan Wehmeir
- Dr. Tanya Khovanova
- The PRIMES Program
- My Family