BILL START RECORDING

Pollard's ρ Algorithm for Factoring (1975)

Thought Experiment

We want to factor N.

Thought Experiment

We want to factor N.

p is a factor of N (we don't know p). Note $p \leq N^{1/2}$.

Thought Experiment

We want to factor N.

p is a factor of N (we don't know p). Note $p \le N^{1/2}$.

We **somehow** find x, y such that $x \equiv y \pmod{p}$. Useful?

Thought Experiment

We want to factor N.

p is a factor of N (we don't know p). Note $p \leq N^{1/2}$.

We **somehow** find x, y such that $x \equiv y \pmod{p}$. Useful?

$\gcd(x - y, N)$ will likely yield a nontrivial factor of N since p divides both.

Thought Experiment

We want to factor N.

p is a factor of N (we don't know p). Note $p \le N^{1/2}$.

We **somehow** find x, y such that $x \equiv y \pmod{p}$. Useful?

$\gcd(x - y, N)$ will likely yield a nontrivial factor of N since p divides both.

We look at several approaches to finding such an x, y that do not work before presenting the approach that does work.

Approach 1: Rand Seq mod p, Intuition

Generate random sequence $x_1, x_2, \ldots \in \{0, \ldots, N-1\}$.

Approach 1: Rand Seq mod p, Intuition

Generate random sequence $x_1, x_2, \ldots \in \{0, \ldots, N-1\}$.

Every time you get a new x_i, test, for all $1 \le j \le i-1$,

$$x_i \equiv x_j \pmod{p}.$$

Approach 1: Rand Seq mod p, Intuition

Generate random sequence $x_1, x_2, \ldots \in \{0, \ldots, N-1\}$.

Every time you get a new x_i, test, for all $1 \le j \le i-1$,

$$x_i \equiv x_j \pmod{p}.$$

Hope to get a YES.

Approach 1: Rand Seq mod p, Intuition

Generate random sequence $x_1, x_2, \ldots \in \{0, \ldots, N-1\}$.

Every time you get a new x_i, test, for all $1 \le j \le i-1$,

$$x_i \equiv x_j \pmod{p}.$$

Hope to get a YES.

If get YES then do

$$\gcd(x_i - x_j, N).$$

Approach 1: Rand Seq mod p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
        for $j \leftarrow 1$ to $i-1$
            if $x_i \equiv x_j \pmod{p}$ then
                $d \leftarrow \gcd(x_i - x_j, N)$
                if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

Approach 1: Rand Seq mod p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
        for $j \leftarrow 1$ to $i-1$
            if $x_i \equiv x_j \pmod{p}$ then
                $d \leftarrow \gcd(x_i - x_j, N)$
                if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's are balls, mod p are boxes. So likely to find $x_i \equiv x_j \pmod{p}$ within $p^{1/2} \sim N^{1/4}$ iterations.

Approach 1: Rand Seq mod p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
        for $j \leftarrow 1$ to $i-1$
            if $x_i \equiv x_j \pmod{p}$ then
                $d \leftarrow \gcd(x_i - x_j, N)$
                if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's are balls, mod p are boxes. So likely to find $x_i \equiv x_j \pmod{p}$ within $p^{1/2} \sim N^{1/4}$ iterations.

CON: Need to already know p.

Approach 1: Rand Seq mod p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
        for $j \leftarrow 1$ to $i-1$
            if $x_i \equiv x_j \pmod{p}$ then
                $d \leftarrow \gcd(x_i - x_j, N)$
                if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's are balls, mod p are boxes. So likely to find $x_i \equiv x_j \pmod{p}$ within $p^{1/2} \sim N^{1/4}$ iterations.

CON: Need to already know p. Darn!

Approach 1: Rand Seq mod p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
        for $j \leftarrow 1$ to $i-1$
            if $x_i \equiv x_j \pmod{p}$ then
                $d \leftarrow \gcd(x_i - x_j, N)$
                if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's are balls, mod p are boxes. So likely to find $x_i \equiv x_j \pmod{p}$ within $p^{1/2} \sim N^{1/4}$ iterations.

CON: Need to already know p. Darn!

ADJUST: Always do GCD.

Approach 2: Rand Seq mod p, W/O p, Intuition

Generate random sequence $x_1, x_2, \ldots \in \{0, \ldots, N-1\}$.

Every time you get a new x_i, do, for all $1 \le j \le i-1$,

$$\gcd(x_i - x_j, N).$$

So do not need to know p. And if $x_i \equiv x_j \pmod{p}$, you'll get a factor.

Approach 2: Rand Seq mod p, W/O p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$ $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
    for $j \leftarrow 1$ to $i-1$
        $d = \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

Approach 2: Rand Seq mod p, W/O p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$ $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
    for $j \leftarrow 1$ to $i-1$
        $d = \mathrm{gcd}(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's:balls, mod p:boxes. Prob find $x_i \equiv x_j \pmod{p}$ with $i \leq p^{1/2} \sim N^{1/4}$. Perhaps sooner–other prime factors. **Not knowing p does not matter.**

Approach 2: Rand Seq mod p, W/O p, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$ $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
    for $j \leftarrow 1$ to $i-1$
        $d = \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO: Bday paradox: x_i's:balls, mod p:boxes. Prob find $x_i \equiv x_j \pmod{p}$ with $i \le p^{1/2} \sim N^{1/4}$. Perhaps sooner–other prime factors. **Not knowing p does not matter.**

CON: Iteration i makes i^2 operations. Total number of operations:

$$\sum_{i=1}^{N^{1/4}} i^2 \sim (N^{1/4})^3 \sim N^{3/4} \text{ BAD :-(.}$$

Another Issue: Space

$x_1 \leftarrow \mathrm{rand}(1, N-1)$ $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
    for $j \leftarrow 1$ to $i-1$
        $d = \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
        $i \leftarrow i + 1$
output(d)

Another Issue: Space

$x_1 \leftarrow \mathrm{rand}(1, N-1)$ $i \leftarrow 2$
while TRUE
    $x_i \leftarrow \mathrm{rand}(1, N-1)$
    for $j \leftarrow 1$ to $i-1$
        $d = \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
        $i \leftarrow i+1$
output(d)

CON: After Iteration i need to store $x_1, \ldots, x_i$. Since $\sim N^{1/4}$ iterations this is $N^{1/4}$ space. Too much space :-(

Approach 3: Rand Looking Sequence, Intuition

How to create a **random looking** sequence?

Approach 3: Rand Looking Sequence, Intuition

How to create a **random looking** sequence?

- Pick random $x_1, c \in \{1, \ldots, N-1\}$.

Approach 3: Rand Looking Sequence, Intuition

How to create a **random looking** sequence?

- Pick random $x_1, c \in \{1, \ldots, N-1\}$.
- If know x_{i-1}, create

$$x_i = x_{i-1} * x_{i-1} + c \pmod{N}.$$

Approach 3: Rand Looking Sequence, Intuition

How to create a **random looking** sequence?

- Pick random $x_1, c \in \{1, \ldots, N-1\}$.
- If know x_{i-1}, create

$$x_i = x_{i-1} * x_{i-1} + c \pmod{N}.$$

- The sequence x_1, x_2, x_3 will **hopefully** be random enough that the bday paradox applies. We use the informal term **random looking** for this.

Approach 3: Rand Looking Sequence, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow x_{i-1} * x_{i-1} + c \pmod{N}$
    for $j \leftarrow 2$ to $i-1$
    $x_j \leftarrow x_{j-1} * x_{j-1} + c$
        $d \leftarrow \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i + 1$
output(d)

Approach 3: Rand Looking Sequence, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow x_{i-1} * x_{i-1} + c \pmod N$
    for $j \leftarrow 2$ to $i-1$
    $x_j \leftarrow x_{j-1} * x_{j-1} + c$
        $d \leftarrow \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO Empirically seq x_1, x_2 is random enough, so $N^{1/4}$ iterations.

Approach 3: Rand Looking Sequence, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow x_{i-1} * x_{i-1} + c \pmod{N}$
    for $j \leftarrow 2$ to $i-1$
    $x_j \leftarrow x_{j-1} * x_{j-1} + c$
        $d \leftarrow \mathrm{gcd}(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i + 1$
output(d)

PRO Empirically seq x_1, x_2 is random enough, so $N^{1/4}$ iterations.

PRO Space not a problem.

Approach 3: Rand Looking Sequence, Program

$x_1 \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $i \leftarrow 2$
while TRUE
    $x_i \leftarrow x_{i-1} * x_{i-1} + c \pmod{N}$
    for $j \leftarrow 2$ to $i-1$
    $x_j \leftarrow x_{j-1} * x_{j-1} + c$
        $d \leftarrow \gcd(x_i - x_j, N)$
        if $d \neq 1$ and $d \neq N$ then break
    $i \leftarrow i+1$
output(d)

PRO Empirically seq x_1, x_2 is random enough, so $N^{1/4}$ iterations.
PRO Space not a problem.
CON Time still a problem :-(

What Do We Really Want?

We want to find $i, j \leq N^{1/4}$ such that $x_i \equiv x_j \pmod{p}$.

What Do We Really Want?

We want to find $i, j \leq N^{1/4}$ such that $x_i \equiv x_j \pmod{p}$.

Key x_i computed via recurrence so $x_i = x_j \implies x_{i+a} = x_{j+a}$.

What Do We Really Want?

We want to find $i, j \leq N^{1/4}$ such that $x_i \equiv x_j \pmod{p}$.
Key x_i computed via recurrence so $x_i = x_j \implies x_{i+a} = x_{j+a}$.

Lemma If exists $i < j \leq M$ with $x_i \equiv x_j$ then exists $k \leq M$ such that $x_k \equiv x_{2k}$.

Recap

Rand Looking Sequence x_1, c chosen at random in $\{1, \ldots, N\}$, then $x_i = x_{i-1} * x_{i-1} + c \pmod{N}$.

Recap

Rand Looking Sequence x_1, c chosen at random in $\{1, \ldots, N\}$, then $x_i = x_{i-1} * x_{i-1} + c \pmod{N}$.

We want to find i, j such $x_i \equiv x_j \pmod{p}$.

Recap

Rand Looking Sequence x_1, c chosen at random in $\{1, \ldots, N\}$, then $x_i = x_{i-1} * x_{i-1} + c \pmod{N}$.

We want to find i, j such $x_i \equiv x_j \pmod{p}$.

Don't know p. Really want $\gcd(x_i - x_j, N) \neq 1$.

Recap

Rand Looking Sequence x_1, c chosen at random in $\{1, \dots, N\}$, then $x_i = x_{i-1} * x_{i-1} + c \pmod{N}$.

We want to find i, j such $x_i \equiv x_j \pmod{p}$.

Don't know p. Really want $\gcd(x_i - x_j, N) \neq 1$.

Trying all pairs is too much time.
Important **If** there is a pair **then** there is a pair of form x_i, x_{2i}.

Recap

Rand Looking Sequence x_1, c chosen at random in $\{1, \ldots, N\}$, then $x_i = x_{i-1} * x_{i-1} + c \pmod{N}$.

We want to find i, j such $x_i \equiv x_j \pmod{p}$.

Don't know p. Really want $\gcd(x_i - x_j, N) \neq 1$.

Trying all pairs is too much time.
Important **If** there is a pair **then** there is a pair of form x_i, x_{2i}.

Idea Only try pairs of form (x_i, x_{2i}).

Final Algorithm

Define $f_c(x) \leftarrow x * x + c$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
while TRUE
    $x \leftarrow f_c(x)$
    $y \leftarrow f_c(f_c(y))$
    $d \leftarrow \gcd(x - y, N)$
    if $d \neq 1$ and $d \neq N$ then break
output(d)

Final Algorithm

Define $f_c(x) \leftarrow x * x + c$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
while TRUE
    $x \leftarrow f_c(x)$
    $y \leftarrow f_c(f_c(y))$
    $d \leftarrow \gcd(x - y, N)$
    if $d \neq 1$ and $d \neq N$ then break
output(d)
PRO By Bday Paradox will likely finish in $N^{1/4}$ steps.

Final Algorithm

Define $f_c(x) \leftarrow x * x + c$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
while TRUE
  $x \leftarrow f_c(x)$
  $y \leftarrow f_c(f_c(y))$
  $d \leftarrow \gcd(x - y, N)$
  if $d \neq 1$ and $d \neq N$ then break
output(d)

PRO By Bday Paradox will likely finish in $N^{1/4}$ steps.

CON No real cons, but is $N^{1/4}$ fast enough?

How Good In Practice?

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.
- Called **Pollard's ρ Algorithm** since he set $\rho = j - i$.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.
- Called **Pollard's ρ Algorithm** since he set $\rho = j - i$.
- Why we think $N^{1/4}$: Sequence seems random enough for Bday paradox to work.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.
- Called **Pollard's ρ Algorithm** since he set $\rho = j - i$.
- Why we think $N^{1/4}$: Sequence seems random enough for Bday paradox to work.
- Why still unproven:

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.
- Called **Pollard's ρ Algorithm** since he set $\rho = j - i$.
- Why we think $N^{1/4}$: Sequence seems random enough for Bday paradox to work.
- Why still unproven:
  - Proving that a deterministic sequence is **random enough** is hard to do or even define.

How Good In Practice?

- The Algorithm is GOOD. Variations are GREAT.
- Was used to provide first factorization of $2^{2^8} + 1$.
- In 1975 was fastest algorithm in practice. Not anymore.
- Called **Pollard's ρ Algorithm** since he set $\rho = j - i$.
- Why we think $N^{1/4}$: Sequence seems random enough for Bday paradox to work.
- Why still unproven:
  - Proving that a deterministic sequence is **random enough** is hard to do or even define.
  - Irene, Radhika, and Emily have not worked on it yet.

The Old Saying in Reverse

Typically one hears the following about academic research:

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

Pollard's ρ-algorithm is an example of the converse:

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

Pollard's ρ-algorithm is an example of the converse:

It works in practice, can we make it work in theory?

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

Pollard's ρ-algorithm is an example of the converse:

It works in practice, can we make it work in theory?

Why is it important to learn why it works in theory?

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

Pollard's ρ-algorithm is an example of the converse:

It works in practice, can we make it work in theory?

Why is it important to learn why it works in theory?

1. Make sure it really works. This is low-priority. Hey! It works!

The Old Saying in Reverse

Typically one hears the following about academic research:

It works in theory, can we make it work in practice?

Pollard's ρ-algorithm is an example of the converse:

It works in practice, can we make it work in theory?

Why is it important to learn why it works in theory?

1. Make sure it really works. This is low-priority. Hey! It works!
2. If we know how it works in theory then perhaps can improve it. This is high-priority. Commonly theory and practice work together to improve both.

BILL STOP RECORDING