BILL START RECORDING

An Early Idea on Factoring: Jevons' Number

Jevons' Number

In the 1870s William Stanley Jevons wrote of the difficulty of factoring. We paraphrase Solomon Golomb's paraphrase:

> **Jevons observed that there are many cases where an operation is easy but it's inverse is hard. He mentioned encryption and decryption. He mentioned multiplication and factoring. He anticipated RSA!**

Jevons' Number

In the 1870s William Stanley Jevons wrote of the difficulty of factoring. We paraphrase Solomon Golomb's paraphrase:

> **Jevons observed that there are many cases where an operation is easy but it's inverse is hard. He mentioned encryption and decryption. He mentioned multiplication and factoring. He anticipated RSA!**

Jevons thought factoring was hard (prob correct!) and that a certain number would **never** be factored (wrong!). Here is a quote:

Jevons' Number

In the 1870s William Stanley Jevons wrote of the difficulty of factoring. We paraphrase Solomon Golomb's paraphrase:

> **Jevons observed that there are many cases where an operation is easy but it's inverse is hard. He mentioned encryption and decryption. He mentioned multiplication and factoring. He anticipated RSA!**

Jevons thought factoring was hard (prob correct!) and that a certain number would **never** be factored (wrong!). Here is a quote:

> **Can the reader say what two numbers multiplied together will produce**
>
> $$\mathbf{8,616,460,799}$$
>
> **I think it is unlikely that anyone aside from myself will ever know.**

Golomb's Method to Factor Jevons' Number

$$J = 8,616,460,799$$

We apply a method of Fermat (in the 1600's) to the problem of factoring J.

Golomb's Method to Factor Jevons' Number

$$J = 8,616,460,799$$

We apply a method of Fermat (in the 1600's) to the problem of factoring J.

To factor J find x, y such that

$$J = x^2 - y^2 = (x - y)(x + y)$$

So we must narrow our search for x, y.

Golomb's Method to Factor Jevons' Number

$$J = 8,616,460,799$$

We apply a method of Fermat (in the 1600's) to the problem of factoring J.

To factor J find x, y such that

$$J = x^2 - y^2 = (x-y)(x+y)$$

So we must narrow our search for x, y.
For this Review I won't get into how to do that.

Golomb's Method to Factor Jevons' Number

$$J = 8,616,460,799$$

We apply a method of Fermat (in the 1600's) to the problem of factoring J.

To factor J find x, y such that

$$J = x^2 - y^2 = (x - y)(x + y)$$

So we must narrow our search for x, y.
For this Review I won't get into how to do that.
The idea of finding x, y such that $J = x^2 = y^2$ will come up later in the course.

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.
Conjecture We have the arrogance of hindsight.

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.

Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

What a moron! He should have asked a Number Theorist

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.

Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

What a moron! He should have asked a Number Theorist

What was he going to do, Google **Number Theorist** ?

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.

Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

What a moron! He should have asked a Number Theorist

What was he going to do, Google **Number Theorist** ?

- It's easy for **us** to say

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.

Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

What a moron! He should have asked a Number Theorist

What was he going to do, Google **Number Theorist** ?

- It's easy for **us** to say

What a moron! He should have asked a Babbage or Lovelace

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.
Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

What a moron! He should have asked a Number Theorist

What was he going to do, Google **Number Theorist** ?

- It's easy for **us** to say

What a moron! He should have asked a Babbage or Lovelace

We know about the role of computers to speed up calculations, but it's reasonable it never dawned on him.

My Opinion and a Counterpoint

Conjecture Jevons was arrogant. Likely true.
Conjecture We have the arrogance of hindsight.

- It's easy for **us** to say

  What a moron! He should have asked a Number Theorist

  What was he going to do, Google **Number Theorist** ?

- It's easy for **us** to say

  What a moron! He should have asked a Babbage or Lovelace

  We know about the role of computers to speed up calculations, but it's reasonable it never dawned on him.

- **Conclusion**
  - His arrogance: assumed the world would not change much.
  - Our arrogance: knowing how much the world did change.

Factoring Algorithms

Recall Factoring Algorithm Ground Rules

Recall Factoring Algorithm Ground Rules

- We only consider algorithms that, given N, find a non-trivial factor of N.

Recall Factoring Algorithm Ground Rules

- We only consider algorithms that, given N, find a non-trivial factor of N.
- We measure the run time as a function of $\lg N$ which is the **length** of the input. We may use L for this.

Recall Factoring Algorithm Ground Rules

- We only consider algorithms that, given N, find a non-trivial factor of N.
- We measure the run time as a function of $\lg N$ which is the **length** of the input. We may use L for this.
- We count $+$, $-$, $\times$, $\div$ as ONE step. A more refined analysis would count them as $(\lg x)^2$ steps where x is the largest number you are dealing with.

Recall Factoring Algorithm Ground Rules

- We only consider algorithms that, given N, find a non-trivial factor of N.
- We measure the run time as a function of $\lg N$ which is the **length** of the input. We may use L for this.
- We count $+$, $-$, $\times$, $\div$ as ONE step. A more refined analysis would count them as $(\lg x)^2$ steps where x is the largest number you are dealing with.
- We leave out the O-of but always mean O-of

Recall Factoring Algorithm Ground Rules

- We only consider algorithms that, given N, find a non-trivial factor of N.
- We measure the run time as a function of $\lg N$ which is the **length** of the input. We may use L for this.
- We count $+$, $-$, $\times$, $\div$ as ONE step. A more refined analysis would count them as $(\lg x)^2$ steps where x is the largest number you are dealing with.
- We leave out the O-of but always mean O-of
- We leave out the *expected time* but always mean it. Our algorithms are randomized.

Easy Factoring Algorithm

Easy Factoring Algorithm

1. Input(N)

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.
How Much Better? Ignoring (1) constants, (2) the lack of proofs of the runtimes, and (3) cheating a byte, we have:

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.
How Much Better? Ignoring (1) constants, (2) the lack of proofs of the runtimes, and (3) cheating a byte, we have:

- Easy: $N^{1/2} = 2^{L/2}$.

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.
How Much Better? Ignoring (1) constants, (2) the lack of proofs of the runtimes, and (3) cheating a byte, we have:

- Easy: $N^{1/2} = 2^{L/2}$.
- Pollard-Rho Algorithm: $N^{1/4} = 2^{L/4}$.

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.
How Much Better? Ignoring (1) constants, (2) the lack of proofs of the runtimes, and (3) cheating a byte, we have:

- Easy: $N^{1/2} = 2^{L/2}$.
- Pollard-Rho Algorithm: $N^{1/4} = 2^{L/4}$.
- Quad Sieve: $N^{1/L^{1/2}} = 2^{L^{1/2}}$.

Easy Factoring Algorithm

1. Input(N)
2. For $x = 2$ to $\lfloor N^{1/2} \rfloor$
   If x divides N then return x (and jump out of loop!).

This takes time $N^{1/2} = 2^{L/2}$.

Goal Do much better than time $N^{1/2}$.
How Much Better? Ignoring (1) constants, (2) the lack of proofs of the runtimes, and (3) cheating a byte, we have:

- Easy: $N^{1/2} = 2^{L/2}$.
- Pollard-Rho Algorithm: $N^{1/4} = 2^{L/4}$.
- Quad Sieve: $N^{1/L^{1/2}} = 2^{L^{1/2}}$.
- Number Field Sieve (best known): $N^{1/L^{2/3}} = 2^{L^{1/3}}$.

Pollard ρ-Algorithm

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 $p \leq N^{1/2}$.

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.

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 $\{1, \ldots, N\}$, 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}.

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}).

Pollard ρ Algorithm

Define $f_c(x) \leftarrow x * x + c \pmod{N}$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
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)

Pollard ρ Algorithm

Define $f_c(x) \leftarrow x * x + c \pmod{N}$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
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.

Pollard ρ Algorithm

Define $f_c(x) \leftarrow x * x + c \pmod{N}$

$x \leftarrow \mathrm{rand}(1, N-1)$, $c \leftarrow \mathrm{rand}(1, N-1)$, $y \leftarrow f_c(x)$
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.

Pollard $p - 1$ Algorithms

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm).

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm).
3. So $\gcd(2^{p-1} - 1, 11227)$ divides 11227.

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm).
3. So $\gcd(2^{p-1} - 1, 11227)$ divides 11227.
4. So $\gcd(2^{p-1} - 1 \bmod 11227, 11227)$ divides 11227.

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm).
3. So $\gcd(2^{p-1} - 1, 11227)$ divides 11227.
4. So $\gcd(2^{p-1} - 1 \bmod 11227, 11227)$ divides 11227.

Lets find $\gcd(2^{p-1} - 1 \bmod 11227, 11227)$. Good idea?

Thought Experiment

Want to factor 11227.
If p is a prime factor of 11227:

1. p divides 11227.
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm).
3. So $\gcd(2^{p-1} - 1, 11227)$ divides 11227.
4. So $\gcd(2^{p-1} - 1 \bmod 11227, 11227)$ divides 11227.

Lets find $\gcd(2^{p-1} - 1 \bmod 11227, 11227)$. Good idea?

We do not know p :-(If we did know p we would be done.

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)
3. p divides $2^{k(p-1)} - 1 \bmod 11227$ for any k

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)
3. p divides $2^{k(p-1)} - 1$ mod 11227 for any k
4. Raise 2 to a power that we **hope** has $p - 1$ as a divisor.

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)
3. p divides $2^{k(p-1)} - 1$ mod 11227 for any k
4. Raise 2 to a power that we **hope** has $p - 1$ as a divisor.

$$\gcd(2^{2^3 \times 3^3} - 1 \bmod 11227, 11227) = \gcd(2^{216} - 1 \bmod 11227, 11227)$$

$$= \gcd(1417, 11227) = 109$$

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)
3. p divides $2^{k(p-1)} - 1 \bmod 11227$ for any k
4. Raise 2 to a power that we **hope** has $p - 1$ as a divisor.

$$\gcd(2^{2^3 \times 3^3} - 1 \bmod 11227, 11227) = \gcd(2^{216} - 1 \bmod 11227, 11227)$$

$$= \gcd(1417, 11227) = 109$$

Great! We got a factor of 11227 without having to factor!

Making the Example Work

Want to factor 11227.
If p is a prime factor of 11227. We do not know p.

1. p divides 11227
2. p divides $2^{p-1} - 1$ (this is always true by Fermat's little Thm)
3. p divides $2^{k(p-1)} - 1 \bmod 11227$ for any k
4. Raise 2 to a power that we **hope** has $p - 1$ as a divisor.

$$\gcd(2^{2^3 \times 3^3} - 1 \bmod 11227, 11227) = \gcd(2^{216} - 1 \bmod 11227, 11227)$$

$$= \gcd(1417, 11227) = 109$$

Great! We got a factor of 11227 without having to factor!
Why Worked 109 was a factor and $108 = 2^2 \times 3^3$, small factors.

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).
- p divides N (our hypothesis).

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).
- p divides N (our hypothesis).
- Hence $\gcd(a^{p-1} - 1 \bmod N, N)$ will be a factor of N.

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).
- p divides N (our hypothesis).
- Hence $\gcd(a^{p-1} - 1 \bmod N, N)$ will be a factor of N.

Two problems:

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod{p}$.

Idea $a^{p-1} - 1 \equiv 0 \pmod{p}$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).
- p divides N (our hypothesis).
- Hence $\gcd(a^{p-1} - 1 \text{ mod } N, N)$ will be a factor of N.

Two problems:

- The GCD might be 1 or N. Thats okay- we can try another a.

General Idea

Fermat's Little Theorem If p is prime and a is coprime to p then $a^{p-1} \equiv 1 \pmod p$.

Idea $a^{p-1} - 1 \equiv 0 \pmod p$. Pick an a at random. If p is a factor of N then:

- p divides $a^{p-1} - 1$ (always).
- p divides N (our hypothesis).
- Hence $\gcd(a^{p-1} - 1 \bmod N, N)$ will be a factor of N.

Two problems:

- The GCD might be 1 or N. Thats okay- we can try another a.
- **We don't have p.** If we did, we'd be done!

Do You Believe in Hope ?

$a^{p-1} \equiv 1 \pmod{p}$. So for all k, $a^{k(p-1)} \equiv 1 \pmod{p}$.

Do You Believe in **Hope** ?

$a^{p-1} \equiv 1 \pmod{p}$. So for all k, $a^{k(p-1)} \equiv 1 \pmod{p}$.

Idea Let M be a number with LOTS of factors.

Do You Believe in Hope ?

$a^{p-1} \equiv 1 \pmod{p}$. So for all k, $a^{k(p-1)} \equiv 1 \pmod{p}$.
Idea Let M be a number with LOTS of factors.
Hope $p-1$ is a factor of M.

Example of B, M

Let B be a parameter.

Example of B, M

Let B be a parameter.

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

Example of B, M

Let B be a parameter.

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$

Example of B, M

Let B be a parameter.

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$

$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.

Example of B, M

Let B be a parameter.

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.

Example of B, M

Let B be a parameter.

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.
$q = 5$, $\lceil \log_5(10) \rceil = 2$. So 5^2.

Example of B, M

Let B be a parameter.

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.
$q = 5$, $\lceil \log_5(10) \rceil = 2$. So 5^2.
$q = 7$, $\lceil \log_7(10) \rceil = 2$. So 7^2.

Example of B, M

Let B be a parameter.

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.
$q = 5$, $\lceil \log_5(10) \rceil = 2$. So 5^2.
$q = 7$, $\lceil \log_7(10) \rceil = 2$. So 7^2.

$$M = 2^4 \times 3^4 \times 5^2 \times 7^2$$

Example of B, M

Let B be a parameter.

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.
$q = 5$, $\lceil \log_5(10) \rceil = 2$. So 5^2.
$q = 7$, $\lceil \log_7(10) \rceil = 2$. So 7^2.

$$M = 2^4 \times 3^4 \times 5^2 \times 7^2$$

If $p - 1 = 2^w 3^x 5^y 7^z$ where $0 \le w, x \le 4$, $0 \le y, z \le 2$ then

Example of B, M

Let B be a parameter.

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

If $B = 10$
$q = 2$, $\lceil \log_2(10) \rceil = 3$. So 2^3.
$q = 3$, $\lceil \log_3(10) \rceil = 4$. So 3^4.
$q = 5$, $\lceil \log_5(10) \rceil = 2$. So 5^2.
$q = 7$, $\lceil \log_7(10) \rceil = 2$. So 7^2.

$$M = 2^4 \times 3^4 \times 5^2 \times 7^2$$

If $p - 1 = 2^w 3^x 5^y 7^z$ where $0 \leq w, x \leq 4$, $0 \leq y, z \leq 2$ then

$$\gcd(a^M - 1, N) \text{ will be a multiple of } p.$$

Do You Believe in Hope ? The Algorithm

Parameter B and hence also

$$M = \prod_{q \leq B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

Do You Believe in Hope ? The Algorithm

Parameter B and hence also

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

```
FOUND = FALSE
while NOT FOUND
      a=RAND(1,N-1)
      d=GCD(a^M-1,N)
      if d=1 then increase B
      if d=N then decrease B
      if (d NE 1) and (d NE N) then FOUND=TRUE
output(d)
```

Do You Believe in Hope ? The Algorithm

Parameter B and hence also

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

```
FOUND = FALSE
while NOT FOUND
     a=RAND(1,N-1)
     d=GCD(a^M-1,N)
     if d=1 then increase B
     if d=N then decrease B
     if (d NE 1) and (d NE N) then FOUND=TRUE
output(d)
```

FACT If $p-1$ has all factors $\le B$ then runtime is $B \log B (\log N)^2$.

Do You Believe in Hope ? The Algorithm

Parameter B and hence also

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

```
FOUND = FALSE
while NOT FOUND
     a=RAND(1,N-1)
     d=GCD(a^M-1,N)
     if d=1 then increase B
     if d=N then decrease B
     if (d NE 1) and (d NE N) then FOUND=TRUE
output(d)
```

FACT If $p-1$ has all factors $\le B$ then runtime is $B \log B (\log N)^2$.
FACT B big then runtime Bad but prob works.

Do You Believe in Hope ? The Algorithm

Parameter B and hence also

$$M = \prod_{q \le B, q \text{ prime}} q^{\lceil \log_q(B) \rceil}.$$

```
FOUND = FALSE
while NOT FOUND
     a=RAND(1,N-1)
     d=GCD(a^M-1,N)
     if d=1 then increase B
     if d=N then decrease B
     if (d NE 1) and (d NE N) then FOUND=TRUE
output(d)
```

FACT If $p-1$ has all factors $\le B$ then runtime is $B \log B (\log N)^2$.
FACT B big then runtime Bad but prob works.
FACT Works well if $p-1$ only has small factors.

In Practice

A rule-of-thumb in practice is to take $B \sim N^{1/6}$.

In Practice

A rule-of-thumb in practice is to take $B \sim N^{1/6}$.

1. Fairly big so the M will be big enough.

In Practice

A rule-of-thumb in practice is to take $B \sim N^{1/6}$.

1. Fairly big so the M will be big enough.
2. Run time $N^{1/6}(\log N)^3$ pretty good, though still exp in $\log N$.

In Practice

A rule-of-thumb in practice is to take $B \sim N^{1/6}$.

1. Fairly big so the M will be big enough.
2. Run time $N^{1/6}(\log N)^3$ pretty good, though still exp in $\log N$.
3. **Warning** This **does not** mean we have an $N^{1/6}(\log N)^3$ algorithm for factoring. It only means we have that if $p-1$ has all factors $\leq N^{1/6}$.

Advice for Alice and Bob

Advice for Alice and Bob

1. Want p, q primes such that $p - 1$ and $q - 1$ have some large factors.

Advice for Alice and Bob

1. Want p, q primes such that $p - 1$ and $q - 1$ have some large factors.
2. Do we know a way to make sure that $p - 1$ and $q - 1$ have some large factors?

Advice for Alice and Bob

1. Want p, q primes such that $p-1$ and $q-1$ have some large factors.
2. Do we know a way to make sure that $p-1$ and $q-1$ have some large factors?
3. Make p, q **safe primes** . Then $p-1=2r$ where r is prime, and $q-1=2s$ where s is prime.

Advice for Alice and Bob

1. Want p, q primes such that $p-1$ and $q-1$ have some large factors.
2. Do we know a way to make sure that $p-1$ and $q-1$ have some large factors?
3. Make p, q **safe primes** . Then $p-1=2r$ where r is prime, and $q-1=2s$ where s is prime.

The usual lesson, so I sound like a broken record, not that your generation knows what a broken record sounds like or even is Because of Pollard's $p-1$ algorithm, Alice and Bob need to use safe primes. A new way to **up their game** .

BILL STOP RECORDING