BILL AND NATHAN, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Factoring
Is Probably Not NPC

BILL START RECORDING

Factoring: Some History

Jevons' Number

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

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

$$\mathbf{8,616,460,799}$$

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.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?

Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?

Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.
5. Golomb in 1996 showed that, given the math **of his day,** Jevons' number could be factored by hand.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.
5. Golomb in 1996 showed that, given the math **of his day,** Jevons' number could be factored by hand.
6. **Student:** Why didn't Jevons just Google **Factoring Quickly**

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.
5. Golomb in 1996 showed that, given the math **of his day,** Jevons' number could be factored by hand.
6. **Student:** Why didn't Jevons just Google **Factoring Quickly**
   Bill: They didn't have the Web back then. Or Google.

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.
5. Golomb in 1996 showed that, given the math **of his day,** Jevons' number could be factored by hand.
6. **Student:** Why didn't Jevons just Google **Factoring Quickly**
   Bill: They didn't have the Web back then. Or Google.
   Student: How did they live?

Jevons' Number

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

We can now factor J easily. Was Jevons' comment stupid?
Discuss

1. Jevons lived 1835–1882 (Died at 46, drowned while swimming.)
2. Jevons did not predict computers. Should he have?
3. Jevons did not predict math would help. Should he have?
4. Lehmer factored J in 1903 using math and computation.
5. Golomb in 1996 showed that, given the math **of his day,** Jevons' number could be factored by hand.
6. **Student:** Why didn't Jevons just Google **Factoring Quickly**
   Bill: They didn't have the Web back then. Or Google.
   Student: How did they live?
   Bill: How indeed!

Was Jevons Arrogant?

Conjecture Jevons was arrogant. Likely true.

Was Jevons Arrogant?

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

Was Jevons Arrogant?

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

Was Jevons Arrogant?

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** ?

Was Jevons Arrogant?

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

Was Jevons Arrogant?

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

Was Jevons Arrogant?

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.

Was Jevons Arrogant?

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.

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) allowing randomized algorithms, 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) allowing randomized algorithms, 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) allowing randomized algorithms, 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) allowing randomized algorithms, 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) allowing randomized algorithms, 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): $N^{1/L^{2/3}} = 2^{L^{1/3}}$.

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) allowing randomized algorithms, 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): $N^{1/L^{2/3}} = 2^{L^{1/3}}$.
- New SVP algorithm: Unclear!

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Has Logic been used to obtain fast SAT algorithms?

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Has Logic been used to obtain fast SAT algorithms?
No.

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Has Logic been used to obtain fast SAT algorithms?
No.
There are algorithms for 3-SAT that take $O((1.5)^n)$.

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Has Logic been used to obtain fast SAT algorithms?
No.
There are algorithms for 3-SAT that take $O((1.5)^n)$.
They used cleverness but no hard math.

Number Theory vs SAT

Has Number Theory been used to obtain fast factoring algorithms?
Yes

Has Logic been used to obtain fast SAT algorithms?
No.
There are algorithms for 3-SAT that take $O((1.5)^n)$.
They used cleverness but no hard math.

This is an informal diff between Factoring and SAT.

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \leq a)[b \text{ divides } n]\}$$

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

Note that $\mathrm{FACT} \in \mathrm{NP}$.

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

Note that $\mathrm{FACT} \in \mathrm{NP}$.

Easy to show that $\mathrm{FACT} \in \mathrm{P}$ iff $f \in \mathrm{PF}$.

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

Note that $\mathrm{FACT} \in \mathrm{NP}$.

Easy to show that $\mathrm{FACT} \in \mathrm{P}$ iff $f \in \mathrm{PF}$.

So our questions is: is FACT NPC?

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

Note that $\mathrm{FACT} \in \mathrm{NP}$.

Easy to show that $\mathrm{FACT} \in \mathrm{P}$ iff $f \in \mathrm{PF}$.

So our questions is: is FACT NPC?
We show $\mathrm{FACT} \in \text{co-NP}$.

Factoring as a Set

Factoring is naturally thought of as a function:

$$f(n) = \text{the least prime factor of } n.$$

Here is the set version for purposes of NPC.

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

Note that $\mathrm{FACT} \in \mathrm{NP}$.

Easy to show that $\mathrm{FACT} \in \mathrm{P}$ iff $f \in \mathrm{PF}$.

So our questions is: is FACT NPC?
We show $\mathrm{FACT} \in \text{co-NP}$.
Hence

$$\mathrm{FACT} \text{ NPC} \rightarrow \mathrm{NP} = \text{co-NP}, \text{ which is unlikely.}$$

Primality in NP

What we Know about Primality

$$\text{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

What we Know about Primality

$$\text{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.

What we Know about Primality

$$\text{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.
2. 1976: Miller got ERH implies PRIMES in P.

What we Know about Primality

$$\mathrm{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.
2. 1976: Miller got ERH implies PRIMES in P.
3. 1977: Solovay-Strassen got PRIMES in RP. Real-world Fast!

What we Know about Primality

$$\text{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.
2. 1976: Miller got ERH implies PRIMES in P.
3. 1977: Solovay-Strassen got PRIMES in RP. Real-world Fast!
4. 1980: Rabin got PRIMES in RP. Real-world Fast!

What we Know about Primality

$$\text{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.
2. 1976: Miller got ERH implies PRIMES in P.
3. 1977: Solovay-Strassen got PRIMES in RP. Real-world Fast!
4. 1980: Rabin got PRIMES in RP. Real-world Fast!
5. 2002 Agrawal-Kayal-Saxe got PRIMES in P. Real-world Slow!

What we Know about Primality

$$\mathrm{PRIMES} = \{x : (\forall y, z)[x = yz \rightarrow (y = 1 \vee z = 1)]\} \in \text{co-NP}$$

1. 1975: Von Pratt got PRIMES in NP.
2. 1976: Miller got ERH implies PRIMES in P.
3. 1977: Solovay-Strassen got PRIMES in RP. Real-world Fast!
4. 1980: Rabin got PRIMES in RP. Real-world Fast!
5. 2002 Agrawal-Kayal-Saxe got PRIMES in P. Real-world Slow!

We will present PRIMES in NP and that is all we will need in our proof that $\mathrm{FACT} \in \text{co-NP}$.

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x, y) = 1]\}.$$

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x,y) = 1]\}.$$

The string y has been called

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x, y) = 1]\}.$$

The string y has been called

1. A **proof** that $x \in A$.

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x,y) = 1]\}.$$

The string y has been called

1. A **proof** that $x \in A$.
2. A **certificate** for $x \in A$.

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x, y) = 1]\}.$$

The string y has been called

1. A **proof** that $x \in A$.
2. A **certificate** for $x \in A$.

We will use the term **certificate** since **proof** has a different connotation.

Terminology for NP

Recall that
$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[B(x, y) = 1]\}.$$

The string y has been called

1. A **proof** that $x \in A$.
2. A **certificate** for $x \in A$.

We will use the term **certificate** since **proof** has a different connotation.

We abbreviate **certificate** by **cert**.

Lucas's Theorem

Let $n \in \mathbb{N}$. Assume there exists a such that

Lucas's Theorem

Let $n \in \mathbb{N}$. Assume there exists a such that

1. $a^{n-1} \equiv 1 \pmod{n}$, and

Lucas's Theorem

Let $n \in \mathbb{N}$. Assume there exists a such that

1. $a^{n-1} \equiv 1 \pmod{n}$, and
2. for every factor q of $n-1$, $a^{(n-1)/q} \not\equiv 1 \pmod{n}$,

then n is prime.

Attempt at Primality in NP

The cert for n prime is

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n - 1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

   (The cert included a factorization of $n - 1$ so the verifier knows all of the factors of $n - 1$.)

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

   (The cert included a factorization of $n-1$ so the verifier knows all of the factors of $n-1$.)

Does this work?

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

(The cert included a factorization of $n-1$ so the verifier knows all of the factors of $n-1$.)

Does this work? I said **Attempt at...** so no.

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

   (The cert included a factorization of $n-1$ so the verifier knows all of the factors of $n-1$.)

Does this work? I said **Attempt at...** so no.
The verifier has to verify that the factorization of $n-1$ is a factorization into primes.

Attempt at Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n - 1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

(The cert included a factorization of $n - 1$ so the verifier knows all of the factors of $n - 1$.)

Does this work? I said **Attempt at...** so no.
The verifier has to verify that the factorization of $n - 1$ is a factorization into primes.

So need the cert to contain a cert that the claimed prime factors of $n - 1$ are prime.

Primality in NP

The cert for n prime is

Primality in NP

The cert for n prime is

1. A number a.

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n - 1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

The verifier does the following:

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

3. Check the cert that each p_i is prime.

So it's a recursive cert.

Primality in NP

The cert for n prime is

1. A number a.
2. A factorization of $n-1 = p_1^{c_1} \cdots p_k^{c_k}$ where p_i's are prime.
3. For each p_i give a cert that p_i is prime. (The cert will be a number a_i such that...) and a factorization of $p_i - 1$....

The verifier does the following:

1. Check that $a^{n-1} \equiv 1 \pmod{n}$,
2. Check that every for every factor q of $n-1$,

$$a^{(n-1)/q} \not\equiv 1 \pmod{n}.$$

3. Check the cert that each p_i is prime.

So it's a recursive cert.
Need to check that the cert is short, but this is not difficult.

Back to Factoring

$\overline{\mathrm{FACT}} \in \mathrm{NP}$

$$\mathrm{FACT} = \{(n,a) : (\exists b \le a)[b \text{ divides } n]\}$$

$$\overline{\mathrm{FACT}} = \{(n,a) : (\forall b \le a)[b \text{ does not divides } n]\}$$

$\overline{\mathrm{FACT}} \in \mathrm{NP}$

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

$$\overline{\mathrm{FACT}} = \{(n, a) : (\forall b \le a)[b \text{ does not divides } n]\}$$

Here is cert that $(n, a) \in \overline{\mathrm{FACT}}$.

1. A factorization $n = p_1^{c_1} \cdots p_k^{c_k}$ where $p_1 < \cdots < p_k$.
2. For each p_i, the cert that p_i is prime.

$\overline{\mathrm{FACT}} \in \mathrm{NP}$

$$\mathrm{FACT} = \{(n, a) : (\exists b \le a)[b \text{ divides } n]\}$$

$$\overline{\mathrm{FACT}} = \{(n, a) : (\forall b \le a)[b \text{ does not divides } n]\}$$

Here is cert that $(n, a) \in \overline{\mathrm{FACT}}$.

1. A factorization $n = p_1^{c_1} \cdots p_k^{c_k}$ where $p_1 < \cdots < p_k$.
2. For each p_i, the cert that p_i is prime.

Verifier has to check

1. $n = p_1^{c_1} \cdots p_k^{c_k}$.
2. $a < p_1$.
3. Each p_i is prime.

Recap What We Know

$$\overline{\mathrm{FACT}} \in \mathrm{NP}$$

Recap What We Know

$$\overline{\mathrm{FACT}} \in \mathrm{NP}$$

so

$$\mathrm{FACT} \in \text{co-NP}$$

Recap What We Know

$$\overline{\text{FACT}} \in \text{NP}$$

so

$$\text{FACT} \in \text{co-NP}$$

so
If FACT is NPC then

Recap What We Know

$$\overline{\text{FACT}} \in \text{NP}$$

so

$$\text{FACT} \in \text{co-NP}$$

so

If FACT is NPC then

$$\text{SAT} \le \text{FACT} \in \text{co-NP}$$

so

Recap What We Know

$$\overline{\mathrm{FACT}} \in \mathrm{NP}$$

so

$$\mathrm{FACT} \in \text{co-NP}$$

so
If FACT is NPC then

$$\mathrm{SAT} \le \mathrm{FACT} \in \text{co-NP}$$

so

$$\mathrm{NP} = \text{co-NP}.$$

Recap What We Know

$$\overline{\mathrm{FACT}} \in \mathrm{NP}$$

so

$$\mathrm{FACT} \in \text{co-NP}$$

so
If FACT is NPC then

$$\mathrm{SAT} \leq \mathrm{FACT} \in \text{co-NP}$$

so

$$\mathrm{NP} = \text{co-NP}.$$

Could factoring be in P?

Recap What We Know

$$\overline{\mathrm{FACT}} \in \mathrm{NP}$$

so

$$\mathrm{FACT} \in \text{co-NP}$$

so
If FACT is NPC then

$$\mathrm{SAT} \le \mathrm{FACT} \in \text{co-NP}$$

so

$$\mathrm{NP} = \text{co-NP}.$$

Could factoring be in P?
Next slide.

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln\ln N)^{1-t}}$$

with Q.Sieve using $t = \frac{1}{2}$ and N.F.Sieve using $t = \frac{1}{3}$.
Moreover, any method that uses B-factoring must take this long.

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln \ln N)^{1-t}}$$

with Q.Sieve using $t = \frac{1}{2}$ and N.F.Sieve using $t = \frac{1}{3}$. Moreover, any method that uses B-factoring must take this long.

- No progress since N.F.Sieve in 1988.

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln\ln N)^{1-t}}$$

with Q.Sieve using $t=\frac{1}{2}$ and N.F.Sieve using $t=\frac{1}{3}$. Moreover, any method that uses B-factoring must take this long.

- No progress since N.F.Sieve in 1988.
- My opinion: $e^{c(\ln N)^t(\ln\ln N)^{1-t}}$ is the best you can do ever, though t can be improved.

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln\ln N)^{1-t}}$$

with Q.Sieve using $t = \frac{1}{2}$ and N.F.Sieve using $t = \frac{1}{3}$.
Moreover, any method that uses B-factoring must take this long.

- No progress since N.F.Sieve in 1988.
- My opinion: $e^{c(\ln N)^t(\ln\ln N)^{1-t}}$ is the best you can do ever, though t can be improved.
- Why hasn't t been improved? Wagstaff told me:

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln\ln N)^{1-t}}$$

with Q.Sieve using $t = \frac{1}{2}$ and N.F.Sieve using $t = \frac{1}{3}$.
Moreover, any method that uses B-factoring must take this long.

- No progress since N.F.Sieve in 1988.
- My opinion: $e^{c(\ln N)^t(\ln\ln N)^{1-t}}$ is the best you can do ever, though t can be improved.
- Why hasn't t been improved? Wagstaff told me:
  - We've run out of parameters to optimize.

The Future of Factoring

I paraphrase **The Joy of Factoring** by Wagstaff:
The best factoring algorithms have time complexity of the form

$$e^{c(\ln N)^t(\ln\ln N)^{1-t}}$$

with Q.Sieve using $t=\frac{1}{2}$ and N.F.Sieve using $t=\frac{1}{3}$. Moreover, any method that uses B-factoring must take this long.

- No progress since N.F.Sieve in 1988.
- My opinion: $e^{c(\ln N)^t(\ln\ln N)^{1-t}}$ is the best you can do ever, though t can be improved.
- Why hasn't t been improved? Wagstaff told me:
  - We've run out of parameters to optimize.
  - Anthony, Davin, Erika, Jacob, and Nathan have not yet applied Ramsey theory to this problem.

BILL AND NATHAN STOP RECORDING