INTEGER PROGRAMMING

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
- In contrast, LINEAR PROGRAMMING asks whether a system of linear inequalities with integer coefficients has a *rational* solution.

INTEGER PROGRAMMING Is NP-Complete[a]

- SET COVERING can be expressed by the inequalities $Ax \geq \vec{1}$, $\sum_{i=1}^{n} x_i \leq B$, $0 \leq x_i \leq 1$, where
  - x_i is one if and only if S_i is in the cover.
  - A is the matrix whose columns are the bit vectors of the sets $S_1, S_2, \ldots$.
  - $\vec{1}$ is the vector of 1s.
  - The operations in Ax are standard matrix operations.
- This shows INTEGER PROGRAMMING is NP-hard.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.

---

[a] Karp (1972); Papadimitriou (1981).

Christos Papadimitriou (1949–)

Easier or Harder?[a]

- Adding restrictions on the allowable *problem instances* will not make a problem harder.
  - We are now solving a subset of problem instances or special cases.
  - The INDEPENDENT SET proof (p. 364) and the KNAPSACK proof (p. 417): equally hard.
  - CIRCUIT VALUE to MONOTONE CIRCUIT VALUE (p. 317): equally hard.
  - SAT to 2SAT (p. 344): easier.

[a]Thanks to a lively class discussion on October 29, 2003.

Easier or Harder? (concluded)

- Adding restrictions on the allowable *solutions* (the solution space) may make a problem harder, equally hard, or easier.
- It is problem dependent.
  - MIN CUT to BISECTION WIDTH (p. 392): harder.
  - LINEAR PROGRAMMING to INTEGER PROGRAMMING (p. 434): harder.
  - SAT to NAESAT (equally hard by p. 357) and MAX CUT to MAX BISECTION (p. 390): equally hard.
  - 3-COLORING to 2-COLORING (p. 401): easier.

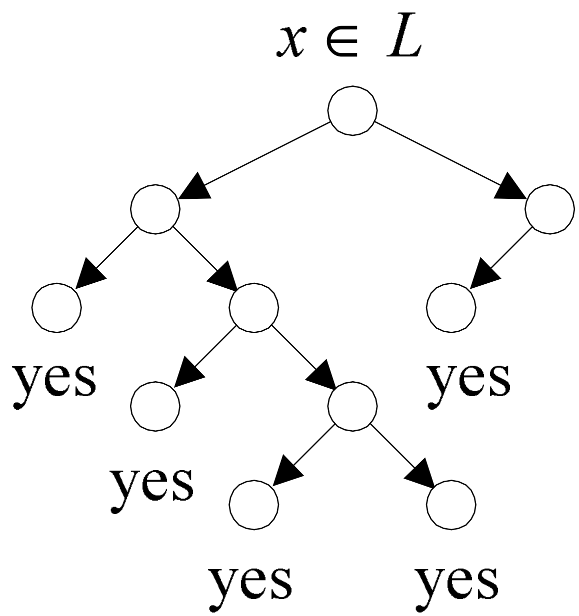
coNP and Function Problems

coNP

- NP is the class of problems that have succinct certificates (recall Proposition 38 on p. 329).
- By definition, coNP is the class of problems whose complement is in NP.
- coNP is therefore the class of problems that have succinct disqualifications:
  - A "no" instance of a problem in coNP possesses a short proof of its being a "no" instance.
  - Only "no" instances have such proofs.

coNP (continued)

- Suppose L is a coNP problem.
- There exists a polynomial-time nondeterministic algorithm M such that:
  - If $x \in L$, then $M(x) =$ "yes" for all computation paths.
  - If $x \notin L$, then $M(x) =$ "no" for some computation path.
- Note that if we swap "yes" and "no" of M, the new algorithm M' decides $\bar{L} \in$ NP in the classic sense (p. 103).


$x \in L$
yes
yes
yes
yes
yes
$x \notin L$
no
yes
yes
no
yes

coNP (continued)

- So there are 3 major approaches to proving $L \in \text{coNP}$.
  1. Prove $\bar{L} \in \text{NP}$.
  2. Prove that only "no" instances possess short proofs.
  3. Write an algorithm for it directly.

coNP (concluded)

- Clearly $\mathrm{P} \subseteq \mathrm{coNP}$.
- It is not known if

$$\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}.$$

  - Contrast this with

$$\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$$

    (see Proposition 14 on p. 169).

Some coNP Problems

- VALIDITY $\in$ coNP.
  - If ϕ is not valid, it can be disqualified very succinctly: a truth assignment that does not satisfy it.
- SAT COMPLEMENT $\in$ coNP.
  - SAT COMPLEMENT is the complement of SAT.
  - The disqualification is a truth assignment that satisfies it.
- HAMILTONIAN PATH COMPLEMENT $\in$ coNP.
  - The disqualification is a Hamiltonian path.

Some coNP Problems (concluded)

- OPTIMAL TSP (D) $\in$ coNP.
  - OPTIMAL TSP (D) asks if the optimal tour has a total distance of B, where B is an input.[a]
  - The disqualification is a tour with a length $< B$.

[a] Defined by Mr. Che-Wei Chang (R95922093) on September 27, 2006.

A Nondeterministic Algorithm for SAT COMPLEMENT (See also p. 113)

ϕ is a boolean formula with n variables.

for $i = 1, 2, \ldots, n$ **do**
    Guess $x_i \in \{0, 1\}$; {Nondeterministic choice.}
end for
{Verification:}
if $\phi(x_1, x_2, \ldots, x_n) = 1$ **then**
    "no";
else
    "yes";
end if

Analysis

- The algorithm decides language $\{\phi : \phi \text{ is unsatisfiable}\}$.
  - The computation tree is a complete binary tree of depth n.
  - Every computation path corresponds to a particular truth assignment out of 2^n.
  - ϕ is unsatisfiable if and only if every truth assignment falsifies ϕ.
  - But every truth assignment falsifies ϕ if and only if every computation path results in "yes."

An Alternative Characterization of coNP

Proposition 50 *Let $L \subseteq \Sigma^*$ be a language. Then $L \in coNP$ if and only if there is a polynomially decidable and polynomially balanced relation R such that*

$$L = \{x : \forall y \, (x, y) \in R\}.$$

(As on p. 328, we assume $|\, y\,| \leq |\, x\,|^k$ for some k.)

- $\bar{L} = \{x : \exists y \, (x, y) \in \neg R\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in$ NP by Proposition 38 (p. 329).
- Hence $L \in$ coNP by definition.

coNP-Completeness

Proposition 51 *L is NP-complete if and only if its complement $\bar{L} = \Sigma^* - L$ is coNP-complete.*

Proof ($\Rightarrow$; the $\Leftarrow$ part is symmetric)

- Let $\bar{L}'$ be any coNP language.
- Hence $L' \in$ NP.
- Let R be the reduction from L' to L.
- So $x \in L'$ if and only if $R(x) \in L$.
- By the law of transposition, $x \notin L'$ if and only if $R(x) \notin L$.

coNP Completeness (concluded)

- So $x \in \bar{L}'$ if and only if $R(x) \in \bar{L}$.
- The *same* R is a reduction from $\bar{L}'$ to $\bar{L}$.
- This shows $\bar{L}$ is coNP-hard.
- But $\bar{L} \in$ coNP.
- This shows $\bar{L}$ is coNP-complete.

Some coNP-Complete Problems

- SAT COMPLEMENT is coNP-complete.
- VALIDITY is coNP-complete.
  - ϕ is valid if and only if $\neg\phi$ is not satisfiable.
  - The reduction from SAT COMPLEMENT to VALIDITY is hence easy.
- HAMILTONIAN PATH COMPLEMENT is coNP-complete.

Possible Relations between P, NP, coNP

1. P = NP = coNP.
2. NP = coNP but P $\neq$ NP.
3. NP $\neq$ coNP and P $\neq$ NP.
   - This is the current "consensus."[a]

---

[a] Carl Gauss (1777–1855), "I could easily lay down a multitude of such propositions, which one could neither prove nor dispose of."

The Primality Problem

- An integer p is **prime** if $p > 1$ and all positive numbers other than 1 and p itself cannot divide it.
- PRIMES asks if an integer N is a prime number.
- Dividing N by $2, 3, \ldots, \sqrt{N}$ is *not* efficient.
  - The length of N is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
  - It is an exponential-time algorithm.
- A polynomial-time algorithm for PRIMES was not found until 2002 by Agrawal, Kayal, and Saxena!
- The running time is $\tilde{O}(\log^{7.5} N)$.

if $n = a^b$ for some $a, b > 1$ **then**
    return "composite";
end if
for $r = 2, 3, \ldots, n-1$ **do**
    if $\gcd(n, r) > 1$ **then**
        return "composite";
    end if
    if r is a prime **then**
        Let q be the largest prime factor of $r - 1$;
        if $q \geq 4\sqrt{r} \log n$ and $n^{(r-1)/q} \neq 1 \bmod r$ **then**
            break; {Exit the for-loop.}
        end if
    end if
end for{$r - 1$ has a prime factor $q \geq 4\sqrt{r} \log n$.}
for $a = 1, 2, \ldots, 2\sqrt{r} \log n$ **do**
    if $(x - a)^n \neq (x^n - a) \bmod (x^r - 1)$ in $Z_n[\,x\,]$ **then**
        return "composite";
    end if
end for
return "prime"; {The only place with "prime" output.}

The Primality Problem (concluded)

- Later, we will focus on efficient "randomized" algorithms for PRIMES (used in *Mathematica*, e.g.).
- $\text{NP} \cap \text{coNP}$ is the class of problems that have succinct certificates and succinct disqualifications.
  - Each "yes" instance has a succinct certificate.
  - Each "no" instance has a succinct disqualification.
  - No instances have both.
- We will see that PRIMES $\in \text{NP} \cap \text{coNP}$.
  - In fact, PRIMES $\in$ P as mentioned earlier.

Primitive Roots in Finite Fields

Theorem 52 (Lucas and Lehmer (1927)) [a] *A number* $p > 1$ *is a prime if and only if there is a number* $1 < r < p$ *such that*

1. $r^{p-1} = 1 \bmod p$, *and*
2. $r^{(p-1)/q} \neq 1 \bmod p$ *for all prime divisors* q *of* $p - 1$.

- This r is called the **primitive root** or **generator**.
- We will prove the theorem later.[b]

---

[a] François Edouard Anatole Lucas (1842–1891); Derrick Henry Lehmer (1905–1991).

[b] See pp. 469ff.

Derrick Lehmer[a] (1905–1991)

[a]Inventor of the linear congruential generator in 1951.

Pratt's Theorem

Theorem 53 (Pratt (1975)) PRIMES $\in NP \cap coNP$.

- PRIMES is in coNP because a succinct disqualification is a proper divisor.
  - A proper divisor of a number n means n is *not* a prime.
- Now suppose p is a prime.
- p's certificate includes the r in Theorem 52 (p. 457).
- Use recursive doubling to check if $r^{p-1} = 1 \bmod p$ in time polynomial in the length of the input, $\log_2 p$.
  - $r, r^2, r^4, \ldots \bmod p$, a total of $\sim \log_2 p$ steps.

The Proof (concluded)

- We also need all *prime* divisors of $p - 1$: $q_1, q_2, \ldots, q_k$.
  - Whether $r, q_1, \ldots, q_k$ are easy to find is irrelevant.
  - There may be multiple choices for r.
- Checking $r^{(p-1)/q_i} \neq 1 \bmod p$ is also easy.
- Checking $q_1, q_2, \ldots, q_k$ are all the divisors of $p - 1$ is easy.
- We still need certificates for the primality of the q_i's.
- The complete certificate is recursive and tree-like:

$$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)). \tag{4}$$

- We next prove that $C(p)$ is succinct.
- As a result, $C(p)$ can be checked in polynomial time.

The Succinctness of the Certificate

Lemma 54 *The length of $C(p)$ is at most quadratic at $5 \log_2^2 p$.*

- This claim holds when $p = 2$ or $p = 3$.
- In general, $p - 1$ has $k \leq \log_2 p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
  - Reason:
$$2^k \leq \prod_{i=1}^{k} q_i \leq p - 1.$$
- Note also that, as $q_1 = 2$,
$$\prod_{i=2}^{k} q_i \leq \frac{p-1}{2}. \tag{5}$$

The Proof (continued)

- $C(p)$ requires:
  - 2 parentheses;
  - $2k < 2\log_2 p$ separators (at most $2\log_2 p$ bits);
  - r (at most $\log_2 p$ bits);
  - $q_1 = 2$ and its certificate 1 (at most 5 bits);
  - $q_2, \ldots, q_k$ (at most $2\log_2 p$ bits);[a]
  - $C(q_2), \ldots, C(q_k)$.

[a] Why?

The Proof (concluded)

- $C(p)$ is succinct because, by induction,

$$
\begin{aligned}
|C(p)| &\leq 5\log_2 p + 5 + 5\sum_{i=2}^{k} \log_2^2 q_i \\
&\leq 5\log_2 p + 5 + 5\left(\sum_{i=2}^{k} \log_2 q_i\right)^2 \\
&\leq 5\log_2 p + 5 + 5\log_2^2 \frac{p-1}{2} \quad \text{by inequality (5)} \\
&< 5\log_2 p + 5 + 5[\,(\log_2 p) - 1\,]^2 \\
&= 5\log_2^2 p + 10 - 5\log_2 p \leq 5\log_2^2 p
\end{aligned}
$$

for $p \geq 4$.

A Certificate for 23[a]

- Note that 5 is a primitive root modulo 23 and $23 - 1 = 22 = 2 \times 11$.[b]
- So
$$C(23) = (5; 2, C(2), 11, C(11)).$$
- Note that 2 is a primitive root modulo 11 and $11 - 1 = 10 = 2 \times 5$.
- So
$$C(11) = (2; 2, C(2), 5, C(5)).$$

[a] Thanks to a lively discussion on April 24, 2008.

[b] Other primitive roots are $7, 10, 11, 14, 15, 17, 19, 20, 21$.

A Certificate for 23 (concluded)

- Note that 2 is a primitive root modulo 5 and $5 - 1 = 4 = 2^2$.
- So
$$C(5) = (2; 2, C(2)).$$
- In summary,
$$C(23) = (5; 2, C(2), 11, (2; 2, C(2), 5, (2; 2, C(2)))).$$
  - In *Mathematica*, `PrimeQCertificate[23]` yields
$$\{23, 5, \{2, \{11, 2, \{2, \{5, 2, \{2\}\}\}\}\}\}$$

Turning the Proof into an Algorithm[a]

- How to turn the proof into a polynomial-time nondeterministic algorithm?
- First, guess a $\log_2 p$-bit number r.
- Then guess up to $\log_2 p$ $\log_2 p$-bit numbers $q_1, q_2, \ldots, q_k$.
- Then recursively do the same thing for each of the q_i to form a certificate (4) on p. 460.
- Finally check if the two conditions of Theorem 52 (p. 457) hold throughout the tree.

[a]Contributed by Mr. Kai-Yuan Hou (`B99201038`, `R03922014`) on November 24, 2015.

Basic Modular Arithmetics[a]

- Let $m, n \in \mathbb{Z}^+$.
- $m \mid n$ means m divides n; m is n's **divisor**.
- We call the numbers $0, 1, \ldots, n-1$ the **residue** modulo n.
- The **greatest common divisor** of m and n is denoted $\gcd(m, n)$.
- The r in Theorem 52 (p. 457) is a primitive root of p.
- We now prove the existence of primitive roots and then Theorem 52 (p. 457).

[a] Carl Friedrich Gauss.

Basic Modular Arithmetics (concluded)

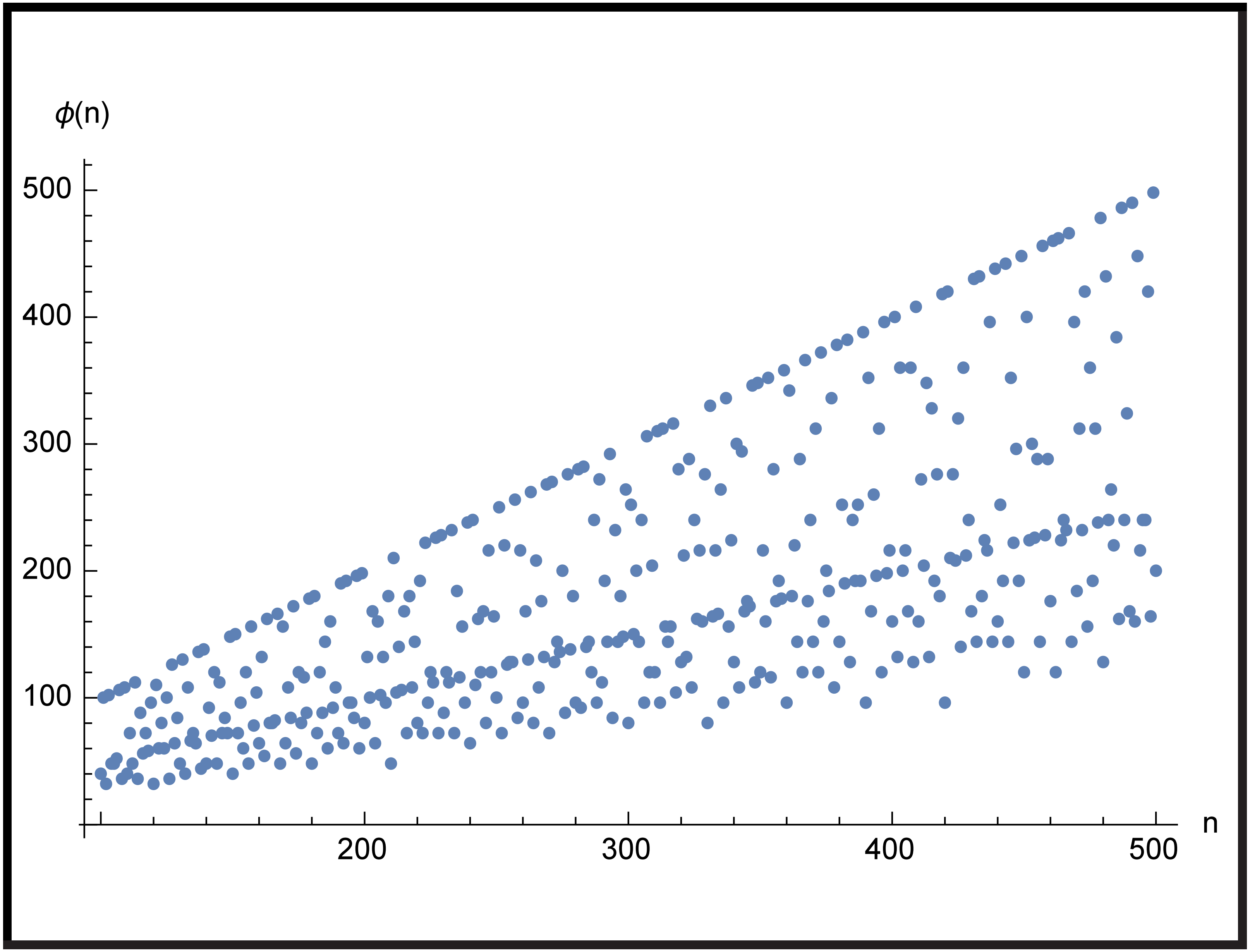
- We use
$$a \equiv b \mod n$$
if $n \,|\, (a - b)$.
  - So $25 \equiv 38 \bmod 13$.
- We use
$$a = b \bmod n$$
if b is the remainder of a divided by n.
  - So $25 = 12 \bmod 13$.

Euler's[a] Totient or Phi Function

- Let
$$\Phi(n) = \{m : 1 \leq m < n, \gcd(m, n) = 1\}$$
be the set of all positive integers less than n that are prime to n.[b]
  - $\Phi(12) = \{1, 5, 7, 11\}$.
- Define **Euler's function** of n to be $\phi(n) = |\,\Phi(n)\,|$.
- $\phi(p) = p - 1$ for prime p, and $\phi(1) = 1$ by convention.
- Euler's function is not expected to be easy to compute without knowing n's factorization.

---

[a] Leonhard Euler (1707–1783).
[b] Z_n^* is an alternative notation.



ϕ(n)
500
400
300
200
100
200
300
400
500
n

Two Properties of Euler's Function

The inclusion-exclusion principle[a] can be used to prove the following.

Lemma 55 $\phi(n) = n \prod_{p \mid n} (1 - \frac{1}{p})$.

- If $n = p_1^{e_1} p_2^{e_2} \cdots p_\ell^{e_\ell}$ is the prime factorization of n, then

$$\phi(n) = n \prod_{i=1}^{\ell} \left(1 - \frac{1}{p_i} \right).$$

Corollary 56 $\phi(mn) = \phi(m)\, \phi(n)$ *if* $\gcd(m, n) = 1$.

[a]Consult any textbooks on discrete mathematics.

A Key Lemma

Lemma 57 $\sum_{m|n} \phi(m) = n$.

- Let $n = \prod_{i=1}^{\ell} p_i^{k_i}$ be the prime factorization of n and consider

$$\prod_{i=1}^{\ell} [\, \phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i})\,]. \tag{6}$$

- Equation (6) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 55 (p. 471) so $\phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i}) = p_i^{k_i}$.
- Expand Eq. (6) to yield

$$n = \sum_{k_1' \leq k_1, \ldots, k_\ell' \leq k_\ell} \prod_{i=1}^{\ell} \phi(p_i^{k_i'}).$$

The Proof (concluded)

- By Corollary 56 (p. 471),

$$\prod_{i=1}^{\ell} \phi(p_i^{k_i'}) = \phi\left(\prod_{i=1}^{\ell} p_i^{k_i'}\right).$$

- So Eq. (6) becomes

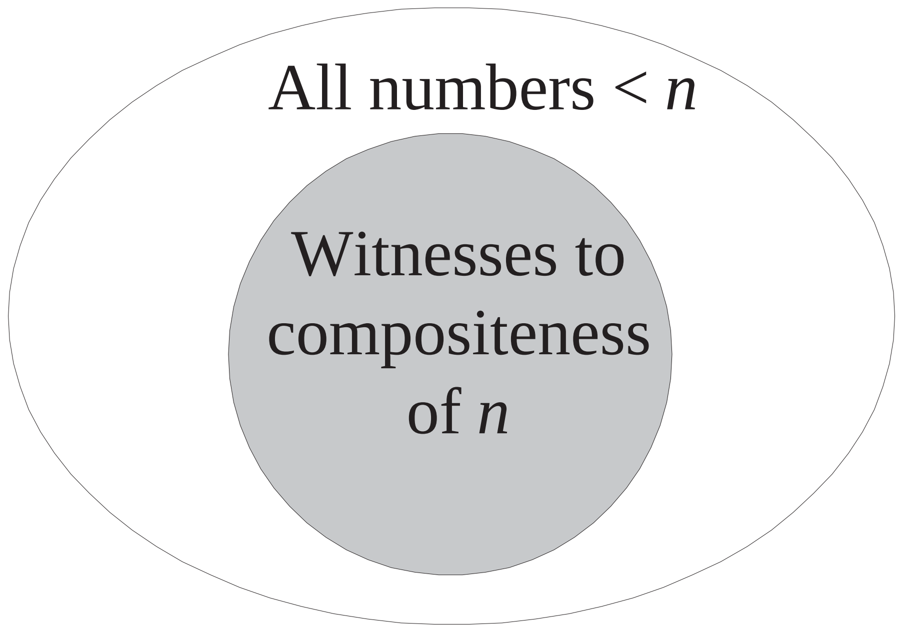
$$n = \sum_{k_1' \leq k_1, \ldots, k_\ell' \leq k_\ell} \phi\left(\prod_{i=1}^{\ell} p_i^{k_i'}\right).$$

- Each $\prod_{i=1}^{\ell} p_i^{k_i'}$ is a unique divisor of $n = \prod_{i=1}^{\ell} p_i^{k_i}$.
- Equation (6) becomes

$$\sum_{m \mid n} \phi(m).$$

Leonhard Euler (1707–1783)

The Density Attack for PRIMES



The Density Attack for PRIMES

Pick $k \in \{1, \ldots, n\}$ randomly;
if $k \mid n$ and $k \neq 1$ and $k \neq n$ **then**
    return "n is composite";
else
    return "n is (probably) a prime";
end if

The Density Attack for PRIMES (continued)

- It works, but does it work well?
- The ratio of numbers $\leq n$ relatively prime to n (the white ring) is
$$\frac{\phi(n)}{n}.$$
- When $n = pq$, where p and q are distinct primes,
$$\frac{\phi(n)}{n} = \frac{pq - p - q + 1}{pq} > 1 - \frac{1}{q} - \frac{1}{p}.$$

The Density Attack for PRIMES (concluded)

- So the ratio of numbers $\leq n$ *not* relatively prime to n (the grey area) is $< (1/q) + (1/p)$.
  - The "density attack" has probability about $2/\sqrt{n}$ of factoring $n = pq$ when $p \sim q = O(\sqrt{n}\,)$.
  - The "density attack" to factor $n = pq$ hence takes $\Omega(\sqrt{n})$ steps on average when $p \sim q = O(\sqrt{n}\,)$.
  - This running time is exponential: $\Omega(2^{0.5 \log_2 n})$.

The Chinese Remainder Theorem

- Let $n = n_1 n_2 \cdots n_k$, where n_i are pairwise relatively prime.
- For any integers $a_1, a_2, \ldots, a_k$, the set of simultaneous equations

$$
\begin{aligned}
x &= a_1 \bmod n_1, \\
x &= a_2 \bmod n_2, \\
&\vdots \\
x &= a_k \bmod n_k,
\end{aligned}
$$

has a unique solution modulo n for the unknown x.

Fermat's "Little" Theorem[a]

Lemma 58 *For all* $0 < a < p$, $a^{p-1} = 1 \bmod p$.

- Recall $\Phi(p) = \{1, 2, \ldots, p-1\}$.
- Consider $a\Phi(p) = \{am \bmod p : m \in \Phi(p)\}$.
- $a\Phi(p) = \Phi(p)$.
  - $a\Phi(p) \subseteq \Phi(p)$ as a remainder must be between 1 and $p-1$.
  - Suppose $am \equiv am' \bmod p$ for $m > m'$, where $m, m' \in \Phi(p)$.
  - That means $a(m - m') = 0 \bmod p$, and p divides a or $m - m'$, which is impossible.

---

[a] Pierre de Fermat (1601–1665).

The Proof (concluded)

- Multiply all the numbers in $\Phi(p)$ to yield $(p-1)!$.
- Multiply all the numbers in $a\Phi(p)$ to yield $a^{p-1}(p-1)!$.
- As $a\Phi(p) = \Phi(p)$, we have
$$a^{p-1}(p-1)! \equiv (p-1)! \bmod p.$$
- Finally, $a^{p-1} = 1 \bmod p$ because $p \not| (p-1)!$.

The Fermat-Euler Theorem[a]

Corollary 59 *For all $a \in \Phi(n)$, $a^{\phi(n)} = 1 \bmod n$.*

- The proof is similar to that of Lemma 58 (p. 480).
- Consider $a\Phi(n) = \{am \bmod n : m \in \Phi(n)\}$.
- $a\Phi(n) = \Phi(n)$.
  - $a\Phi(n) \subseteq \Phi(n)$ as a remainder must be between 0 and $n-1$ and relatively prime to n.
  - Suppose $am \equiv am' \bmod n$ for $m' < m < n$, where $m, m' \in \Phi(n)$.
  - That means $a(m-m') = 0 \bmod n$, and n divides a or $m-m'$, which is impossible.

[a]Proof by Mr. Wei-Cheng Cheng (`R93922108`, `D95922011`) on November 24, 2004.

The Proof (concluded)[a]

- Multiply all the numbers in $\Phi(n)$ to yield $\prod_{m \in \Phi(n)} m$.
- Multiply all the numbers in $a\Phi(n)$ to yield $a^{\phi(n)} \prod_{m \in \Phi(n)} m$.
- As $a\Phi(n) = \Phi(n)$,

$$\prod_{m \in \Phi(n)} m \equiv a^{\phi(n)} \left(\prod_{m \in \Phi(n)} m \right) \bmod n.$$

- Finally, $a^{\phi(n)} = 1 \bmod n$ because $n \not| \prod_{m \in \Phi(n)} m$.

[a]Some typographical errors corrected by Mr. Jung-Ying Chen (D95723006) on November 18, 2008.

An Example

- As $12 = 2^2 \times 3$,

$$\phi(12) = 12 \times \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 4.$$

- In fact, $\Phi(12) = \{1, 5, 7, 11\}$.
- For example,

$$5^4 = 625 = 1 \bmod 12.$$

Exponents

- The **exponent** of $m \in \Phi(p)$ is the least $k \in \mathbb{Z}^+$ such that
$$m^k = 1 \bmod p.$$
- Every residue $s \in \Phi(p)$ has an exponent.
  - $1, s, s^2, s^3, \ldots$ eventually repeats itself modulo p, say $s^i \equiv s^j \bmod p$, which means $s^{j-i} = 1 \bmod p$.
- If the exponent of m is k and $m^\ell = 1 \bmod p$, then $k \mid \ell$.
  - Otherwise, $\ell = qk + a$ for $0 < a < k$, and $m^\ell = m^{qk+a} \equiv m^a \equiv 1 \bmod p$, a contradiction.

Lemma 60 *Any nonzero polynomial of degree k has at most k distinct roots modulo p.*