

Graph Coloring

- k -COLORING asks if the nodes of a graph can be colored with $\leq k$ colors such that no two adjacent nodes have the same color.
- 2-COLORING is in P (why?).
- But 3-COLORING is NP-complete (see next page).
- k -COLORING is NP-complete for $k \geq 3$ (why?).

3-COLORING Is NP-Complete^a

- We will reduce NAESAT to 3-COLORING.
- We are given a set of clauses C_1, C_2, \dots, C_m each with 3 literals.
- The boolean variables are x_1, x_2, \dots, x_n .
- We shall construct a graph G such that it can be colored with colors $\{0, 1, 2\}$ if and only if all the clauses can be NAE-satisfied.

^aKarp (1972).

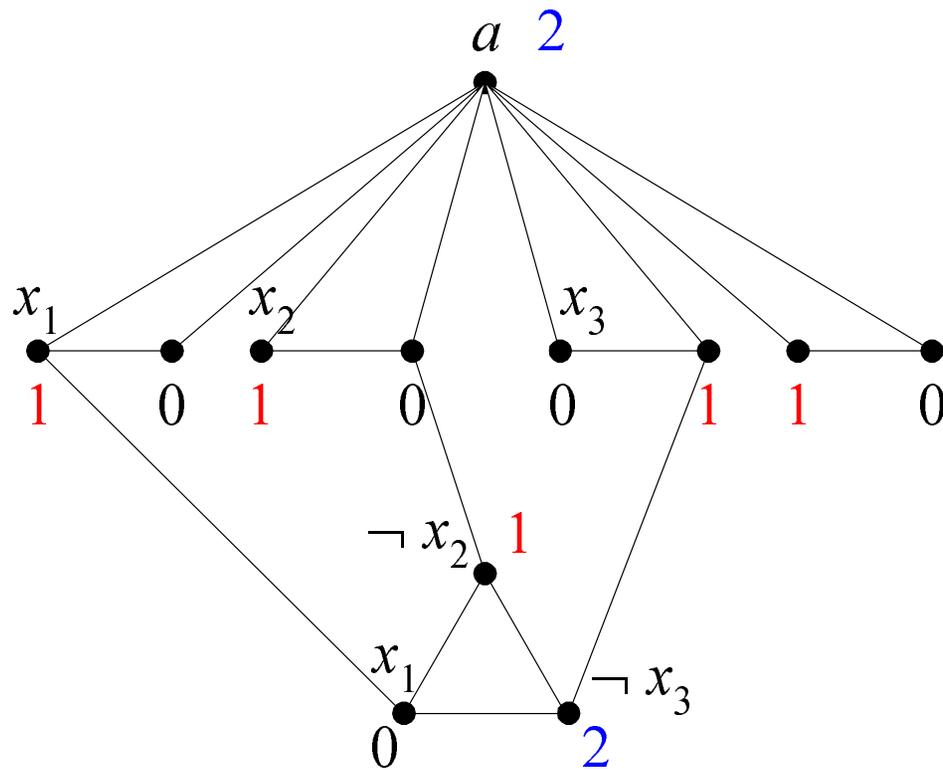
The Proof (continued)

- Every variable x_i is involved in a triangle $[a, x_i, \neg x_i]$ with a common node a .
- Each clause $C_i = (c_{i1} \vee c_{i2} \vee c_{i3})$ is also represented by a triangle

$$[c_{i1}, c_{i2}, c_{i3}].$$

- Node c_{ij} with the same label as one in some triangle $[a, x_k, \neg x_k]$ represent *distinct* nodes.
- There is an edge between c_{ij} and the node that represents the j th literal of C_i .

Construction for $\dots \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \dots$



The Proof (continued)

Suppose the graph is 3-colorable.

- Assume without loss of generality that node a takes the color 2.
- A triangle must use up all 3 colors.
- As a result, one of x_i and $\neg x_i$ must take the color 0 and the other 1.

The Proof (continued)

- Treat 1 as true and 0 as false.^a
 - We were dealing only with those triangles with the a node, not the clause triangles.
- The resulting truth assignment is clearly contradiction free.
- As each clause triangle contains one color 1 and one color 0, the clauses are NAE-satisfied.

^aThe opposite also works.

The Proof (continued)

Suppose the clauses are NAE-satisfiable.

- Color node a with color 2.
- Color the nodes representing literals by their truth values (color 0 for **false** and color 1 for **true**).
 - We were dealing only with those triangles with the a node, not the clause triangles.

The Proof (concluded)

- For each clause triangle:
 - Pick any two literals with opposite truth values.
 - Color the corresponding nodes with 0 if the literal is true and 1 if it is false.
 - Color the remaining node with color 2.
- The coloring is legitimate.
 - If literal w of a clause triangle has color 2, then its color will never be an issue.
 - If literal w of a clause triangle has color 1, then it must be connected up to literal w with color 0.
 - If literal w of a clause triangle has color 0, then it must be connected up to literal w with color 1.

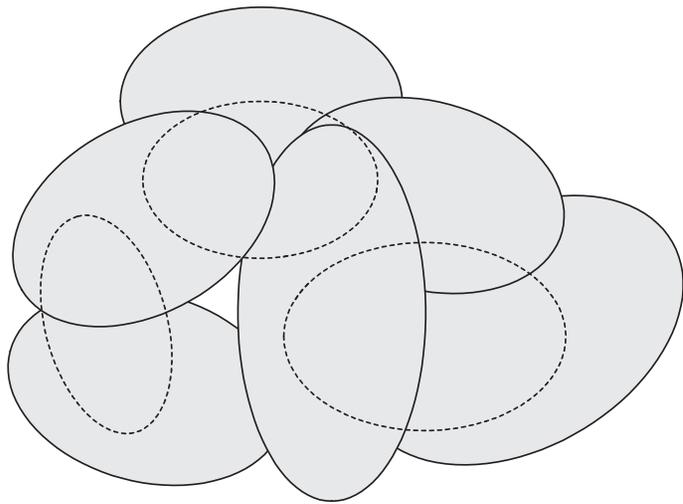
TRIPARTITE MATCHING

- We are given three sets B , G , and H , each containing n elements.
- Let $T \subseteq B \times G \times H$ be a ternary relation.
- TRIPARTITE MATCHING asks if there is a set of n triples in T , none of which has a component in common.
 - Each element in B is matched to a different element in G and different element in H .

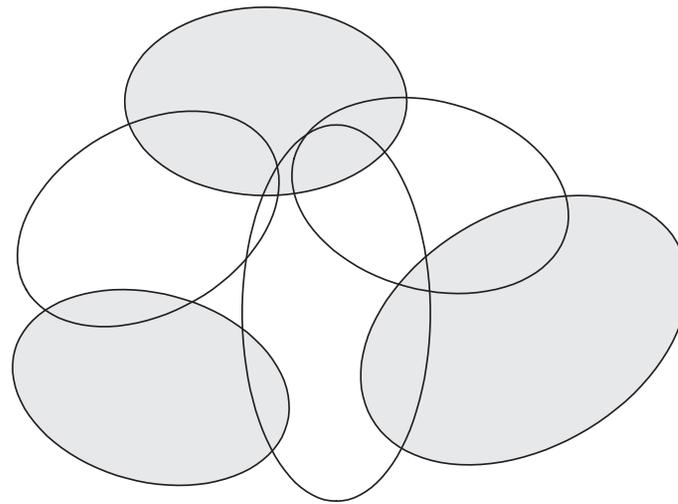
Theorem 39 (Karp (1972)) TRIPARTITE MATCHING *is NP-complete.*

Related Problems

- We are given a family $F = \{S_1, S_2, \dots, S_n\}$ of subsets of a finite set U and a budget B .
- SET COVERING asks if there exists a set of B sets in F whose union is U .
- SET PACKING asks if there are B disjoint sets in F .
- Assume $|U| = 3m$ for some $m \in \mathbb{N}$ and $|S_i| = 3$ for all i .
- EXACT COVER BY 3-SETS asks if there are m sets in F that are disjoint and have U as their union.



SET COVERING



SET PACKING

Related Problems (concluded)

Corollary 40 SET COVERING, SET PACKING, *and* EXACT COVER BY 3-SETS *are all NP-complete.*

The KNAPSACK Problem

- There is a set of n items.
- Item i has value $v_i \in \mathbb{Z}^+$ and weight $w_i \in \mathbb{Z}^+$.
- We are given $K \in \mathbb{Z}^+$ and $W \in \mathbb{Z}^+$.
- KNAPSACK asks if there exists a subset $S \subseteq \{1, 2, \dots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq K$.
 - We want to achieve the maximum satisfaction within the budget.

KNAPSACK Is NP-Complete

- KNAPSACK \in NP: Guess an S and verify the constraints.
- We assume $v_i = w_i$ for all i and $K = W$.
- KNAPSACK now asks if a subset of $\{v_1, v_2, \dots, v_n\}$ adds up to exactly K .
 - Picture yourself as a radio DJ.
 - Or a person trying to control the calories intake.
- We shall reduce EXACT COVER BY 3-SETS to KNAPSACK.

The Proof (continued)

- We are given a family $F = \{S_1, S_2, \dots, S_n\}$ of size-3 subsets of $U = \{1, 2, \dots, 3m\}$.
- EXACT COVER BY 3-SETS asks if there are m disjoint sets in F that cover the set U .
- Think of a set as a bit vector in $\{0, 1\}^{3m}$.
 - 001100010 means the set $\{3, 4, 8\}$, and 110010000 means the set $\{1, 2, 5\}$.
- Our goal is $\overbrace{11 \cdots 1}^{3m}$.

The Proof (continued)

- A bit vector can also be considered as a binary *number*.
- Set union resembles addition.
 - $001100010 + 110010000 = 111110010$, which denotes the set $\{1, 2, 3, 4, 5, 8\}$, as desired.
- Trouble occurs when there is *carry*.
 - $001100010 + 001110000 = 010010010$, which denotes the set $\{2, 5, 8\}$, not the desired $\{3, 4, 5, 8\}$.

The Proof (continued)

- Carry may also lead to a situation where we obtain our solution $11 \cdots 1$ with more than m sets in F .
 - $001100010 + 001110000 + 101100000 + 000001101 = 111111111$.
 - But this “solution” $\{1, 3, 4, 5, 6, 7, 8, 9\}$ does not correspond to an exact cover.
 - And it uses 4 sets instead of the required 3.^a
- To fix this problem, we enlarge the base just enough so that there are no carries.
- Because there are n vectors in total, we change the base from 2 to $n + 1$.

^aThanks to a lively class discussion on November 20, 2002.

The Proof (continued)

- Set v_i to be the $(n + 1)$ -ary number corresponding to the bit vector encoding S_i .
- Now in base $n + 1$, if there is a set S such that

$\sum_{v_i \in S} v_i = \overbrace{11 \cdots 1}^{3m}$, then every bit position must be contributed by exactly one v_i and $|S| = m$.

- Finally, set

$$K = \sum_{j=0}^{3m-1} (n + 1)^j = \overbrace{11 \cdots 1}^{3m} \quad (\text{base } n + 1).$$

The Proof (continued)

- Suppose F admits an exact cover, say $\{S_1, S_2, \dots, S_m\}$.
- Then picking $S = \{v_1, v_2, \dots, v_m\}$ clearly results in

$$v_1 + v_2 + \dots + v_m = \overbrace{11 \dots 1}^{3m}.$$

- It is important to note that the meaning of addition (+) is independent of the base.^a
- It is just regular addition.
- But a S_i may give rise to different v_i 's under different bases.

^aContributed by Mr. Kuan-Yu Chen (R92922047) on November 3, 2004.

The Proof (concluded)

- On the other hand, suppose there exists an S such that

$$\sum_{v_i \in S} v_i = \overbrace{11 \cdots 1}^{3m} \text{ in base } n + 1.$$

- The no-carry property implies that $|S| = m$ and $\{S_i : v_i \in S\}$ is an exact cover.

An Example

- Let $m = 3$, $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and

$$S_1 = \{1, 3, 4\},$$

$$S_2 = \{2, 3, 4\},$$

$$S_3 = \{2, 5, 6\},$$

$$S_4 = \{6, 7, 8\},$$

$$S_5 = \{7, 8, 9\}.$$

- Note that $n = 5$, as there are 5 S_i 's.

An Example (concluded)

- Our reduction produces

$$K = \sum_{j=0}^{3 \times 3 - 1} 6^j = \overbrace{11 \cdots 1}^{3 \times 3} \quad (\text{base } 6) = 2015539,$$

$$v_1 = 101100000 = 1734048,$$

$$v_2 = 011100000 = 334368,$$

$$v_3 = 010011000 = 281448,$$

$$v_4 = 000001110 = 258,$$

$$v_5 = 000000111 = 43.$$

- Note $v_1 + v_3 + v_5 = K$.
- Indeed, $S_1 \cup S_3 \cup S_5 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, an exact cover by 3-sets.

BIN PACKINGS

- We are given N positive integers a_1, a_2, \dots, a_N , an integer C (the capacity), and an integer B (the number of bins).
- BIN PACKING asks if these numbers can be partitioned into B subsets, each of which has total sum at most C .
- Think of packing bags at the check-out counter.

Theorem 41 BIN PACKING *is NP-complete.*

INTEGER PROGRAMMING

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
 - 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, \dots
 - $\vec{1}$ is the vector of 1s.
- This shows INTEGER PROGRAMMING is NP-hard.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.

^aPapadimitriou (1981).

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.
 - The INDEPENDENT SET proof (p. 277) and the KNAPSACK proof (p. 322).
 - SAT to 2SAT (easier by p. 264).
 - CIRCUIT VALUE to MONOTONE CIRCUIT VALUE (equally hard by p. 241).

^aThanks to a lively class discussion on October 29, 2003.

Easier or Harder? (concluded)

- Adding restrictions on the allowable *solutions* may make a problem easier, as hard, or harder.
- It is problem dependent.
 - MIN CUT to BISECTION WIDTH (harder by p. 303).
 - LINEAR PROGRAMMING to INTEGER PROGRAMMING (harder by p. 332).
 - SAT to NAESAT (equally hard by p. 272) and MAX CUT to MAX BISECTION (equally hard by p. 301).
 - 3-COLORING to 2-COLORING (easier by p. 309).

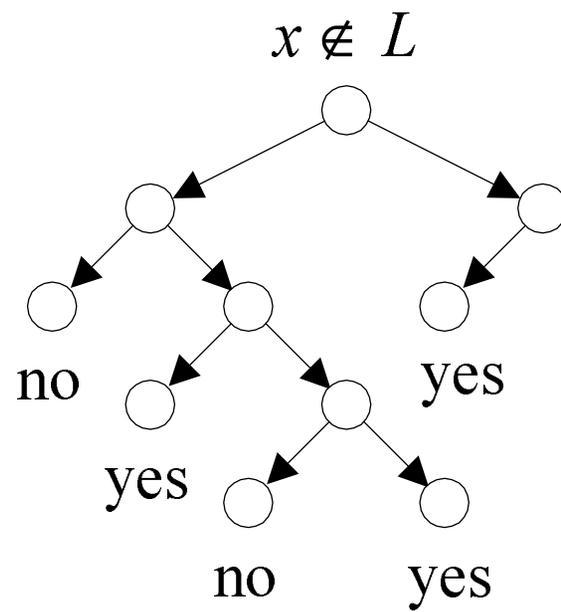
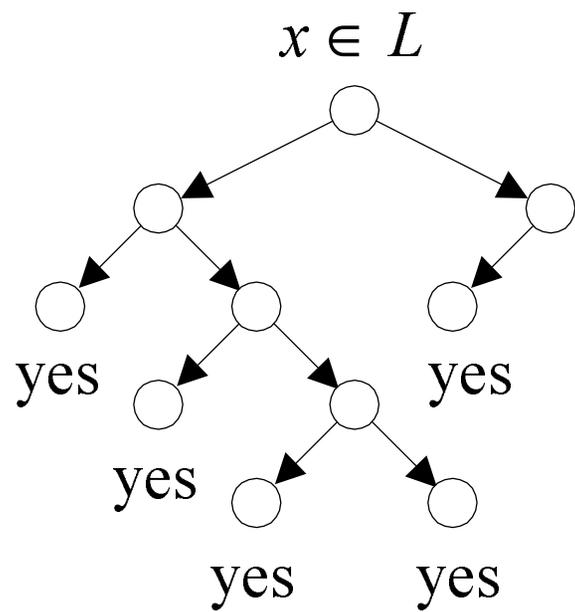
coNP and Function Problems

coNP

- By definition, coNP is the class of problems whose complement is in NP.
- NP is the class of problems that have succinct certificates (recall Proposition 30 on p. 251).
- 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 $M(x) = \text{“yes”}$ for all computation paths.
 - If $x \notin L$, then $M(x) = \text{“no”}$ for some computation path.



coNP (concluded)

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

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

– Contrast this with

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

(see Proposition 11 on p. 124).

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.
 - The disqualification is a truth assignment that satisfies it.
- HAMILTONIAN PATH COMPLEMENT \in coNP.
 - The disqualification is a Hamiltonian path.
- OPTIMAL TSP (D) \in coNP.^a
 - The disqualification is a tour with a length $< B$.

^aAsked by Mr. Che-Wei Chang (R95922093) on September 27, 2006.

An Alternative Characterization of coNP

Proposition 42 *Let $L \subseteq \Sigma^*$ be a language. Then $L \in \text{coNP}$ 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. 250, we assume $|y| \leq |x|^k$ for some k .)

- $\bar{L} = \{x : (x, y) \in \neg R \text{ for some } y\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in \text{NP}$ by Proposition 30 (p. 251).
- Hence $L \in \text{coNP}$ by definition.

coNP Completeness

Proposition 43 *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 $L' \in \text{NP}$.
- Let R be the reduction from L' to L .
- So $x \in L'$ if and only if $R(x) \in L$.
- So $x \in \bar{L}'$ if and only if $R(x) \in \bar{L}$.
- R is a reduction from \bar{L}' to \bar{L} .

Some coNP-Complete Problems

- SAT COMPLEMENT is coNP-complete.
 - SAT COMPLEMENT is the complement of SAT.
- 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. $P = NP = \text{coNP}$.
2. $NP = \text{coNP}$ but $P \neq NP$.
3. $NP \neq \text{coNP}$ and $P \neq NP$.
 - This is current “consensus.”

coNP Hardness and NP Hardness^a

Proposition 44 *If a coNP-hard problem is in NP, then $NP = coNP$.*

- Let $L \in NP$ be coNP-hard.
- Let NTM M decide L .
- For any $L' \in coNP$, there is a reduction R from L' to L .
- $L' \in NP$ as it is decided by NTM $M(R(x))$.
 - Alternatively, NP is closed under complement.
- Hence $coNP \subseteq NP$.
- The other direction $NP \subseteq coNP$ is symmetric.

^aBrassard (1979); Selman (1978).

coNP Hardness and NP Hardness (concluded)

Similarly,

Proposition 45 *If an NP-hard problem is in coNP, then $NP = coNP$.*

Hence NP-complete problems are unlikely to be in coNP and coNP-complete problems are unlikely to be in NP.

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 $2, 3, \dots, \sqrt{N}$ is *not* efficient.
 - The length of N is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
- A polynomial-time algorithm for PRIMES was not found until 2002 by Agrawal, Kayal, and Saxena!
- We will focus on efficient “probabilistic” algorithms for PRIMES (used in *Mathematica*, e.g.).

```

1: if  $n = a^b$  for some  $a, b > 1$  then
2:   return “composite”;
3: end if
4: for  $r = 2, 3, \dots, n - 1$  do
5:   if  $\gcd(n, r) > 1$  then
6:     return “composite”;
7:   end if
8:   if  $r$  is a prime then
9:     Let  $q$  be the largest prime factor of  $r - 1$ ;
10:    if  $q \geq 4\sqrt{r} \log n$  and  $n^{(r-1)/q} \not\equiv 1 \pmod{r}$  then
11:      break; {Exit the for-loop.}
12:    end if
13:  end if
14: end for { $r - 1$  has a prime factor  $q \geq 4\sqrt{r} \log n$ .}
15: for  $a = 1, 2, \dots, 2\sqrt{r} \log n$  do
16:   if  $(x - a)^n \not\equiv (x^n - a) \pmod{(x^r - 1)}$  in  $Z_n[x]$  then
17:     return “composite”;
18:   end if
19: end for
20: return “prime”; {The only place with “prime” output.}

```

DP

- $DP \equiv NP \cap coNP$ 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.
- $P \subseteq DP$.
- We will see that $PRIMES \in DP$.
 - In fact, $PRIMES \in P$ as mentioned earlier.

Primitive Roots in Finite Fields

Theorem 46 (Lucas and Lehmer (1927)) ^a *A number $p > 1$ is prime if and only if there is a number $1 < r < p$ (called the **primitive root** or **generator**) such that*

1. $r^{p-1} = 1 \pmod{p}$, and
 2. $r^{(p-1)/q} \not\equiv 1 \pmod{p}$ for all prime divisors q of $p - 1$.
- We will prove the theorem later.

^aFrançois Edouard Anatole Lucas (1842–1891); Derrick Henry Lehmer (1905–1991).

Pratt's Theorem

Theorem 47 (Pratt (1975)) $\text{PRIMES} \in NP \cap \text{coNP}$.

- PRIMES is in coNP because a succinct disqualification is a divisor.
- Suppose p is a prime.
- p 's certificate includes the r in Theorem 46 (p. 351).
- Use recursive doubling to check if $r^{p-1} = 1 \pmod p$ in time polynomial in the length of the input, $\log_2 p$.
- We also need all *prime* divisors of $p - 1$: q_1, q_2, \dots, q_k .
- Checking $r^{(p-1)/q_i} \neq 1 \pmod p$ is also easy.

The Proof (concluded)

- Checking q_1, q_2, \dots, 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:

$$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \dots, q_k, C(q_k)).$$

- $C(p)$ can also be checked in polynomial time.
- We next prove that $C(p)$ is succinct.

The Succinctness of the Certificate

Lemma 48 *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 $k < \log_2 p$ prime divisors $q_1 = 2, q_2, \dots, q_k$.
- $C(p)$ requires: 2 parentheses and $2k < 2 \log_2 p$ separators (length at most $2 \log_2 p$ long), r (length at most $\log_2 p$), $q_1 = 2$ and its certificate 1 (length at most 5 bits), the q_i 's (length at most $2 \log_2 p$), and the $C(q_i)$ s.

The Proof (concluded)

- $C(p)$ is succinct because

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

for $p \geq 4$.

Basic Modular Arithmetics^a

- Let $m, n \in \mathbb{Z}^+$.
- $m|n$ means m divides n and m is n 's **divisor**.
- We call the numbers $0, 1, \dots, n - 1$ the **residue** modulo n .
- The **greatest common divisor** of m and n is denoted $\gcd(m, n)$.
- The r in Theorem 46 (p. 351) is a primitive root of p .
- We now prove the existence of primitive roots and then Theorem 46.

^aCarl Friedrich Gauss.

Euler's^a Totient or Phi Function

- Let

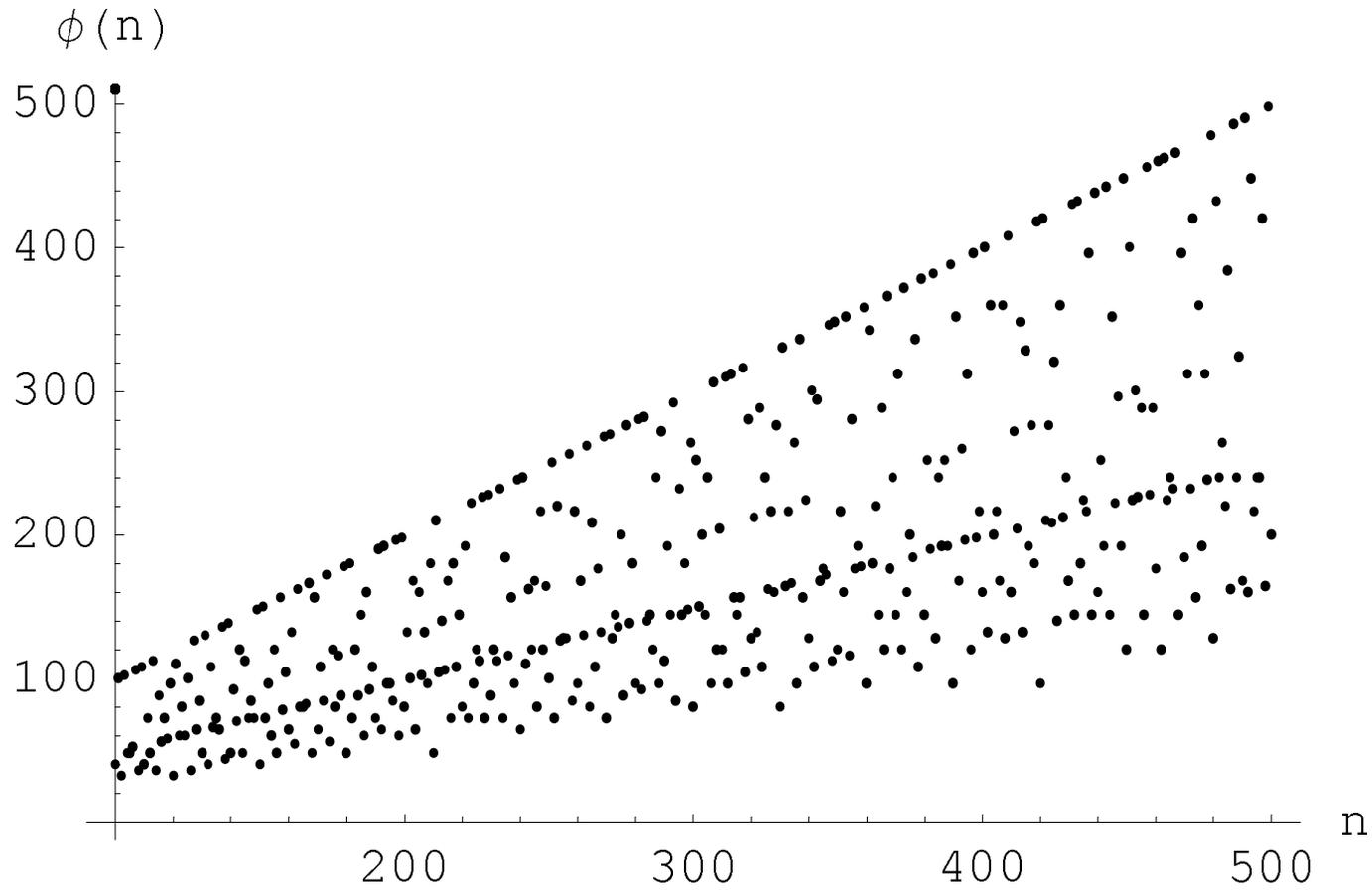
$$\Phi(n) = \{m : 1 \leq m < n, \gcd(m, n) = 1\}$$

be the set of all positive integers less than n that are prime to n (Z_n^* is a more popular notation).

– $\Phi(12) = \{1, 5, 7, 11\}$.

- 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.

^aLeonhard Euler (1707–1783).



Two Properties of Euler's Function

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

Lemma 49 $\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$.

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

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

Corollary 50 $\phi(mn) = \phi(m) \phi(n)$ if $\gcd(m, n) = 1$.

^aSee my *Discrete Mathematics* lecture notes.

A Key Lemma

Lemma 51 $\sum_{m|n} \phi(m) = n.$

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

$$\prod_{i=1}^{\ell} [\phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i})]. \quad (4)$$

- Equation (4) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 49.
- Expand Eq. (4) to yield $\sum_{k'_1 \leq k_1, \dots, k'_\ell \leq k_\ell} \prod_{i=1}^{\ell} \phi(p_i^{k'_i}).$

The Proof (concluded)

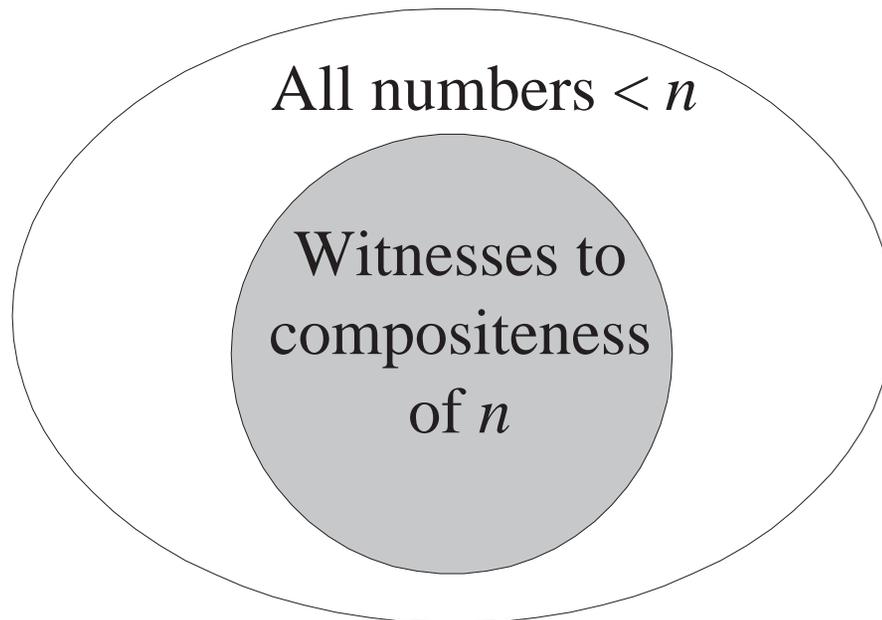
- By Corollary 50 (p. 359),

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

- Each $\prod_{i=1}^{\ell} p_i^{k'_i}$ is a unique divisor of $n = \prod_{i=1}^{\ell} p_i^{k_i}$.
- Equation (4) becomes

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

The Density Attack for PRIMES



- It works, but does it work well?

Factorization and Euler's Function

- The ratio of numbers $\leq n$ relatively prime to n is $\phi(n)/n$.
- 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” 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, \dots, a_k , the set of simultaneous equations

$$x = a_1 \pmod{n_1},$$

$$x = a_2 \pmod{n_2},$$

$$\vdots$$

$$x = a_k \pmod{n_k},$$

has a unique solution modulo n for the unknown x .

Fermat's "Little" Theorem^a

Lemma 52 For all $0 < a < p$, $a^{p-1} = 1 \pmod{p}$.

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

^aPierre 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 $a\Phi(p) = \Phi(p)$, $(p - 1)! = a^{p-1}(p - 1)! \pmod p$.
- Finally, $a^{p-1} = 1 \pmod p$ because $p \nmid (p - 1)!$.

The Fermat-Euler Theorem^a

Corollary 53 For all $a \in \Phi(n)$, $a^{\phi(n)} = 1 \pmod n$.

- The proof is similar to that of Lemma 52 (p. 365).
- Consider $a\Phi(n) = \{am \pmod n : m \in \Phi(n)\}$.
- $a\Phi(n) = \Phi(n)$.
 - $a\Phi(n) \subseteq \Phi(n)$ as a remainder must be between 0 and $n - 1$ and relatively prime to n .
 - Suppose $am = am' \pmod n$ for $m' < m < n$, where $m, m' \in \Phi(n)$.
 - That means $a(m - m') = 0 \pmod n$, and n divides a or $m - m'$, which is impossible.

^aProof by Mr. Wei-Cheng Cheng (R93922108) on November 24, 2004.

The Proof (concluded)

- 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 $a\Phi(n) = \Phi(n)$,

$$\prod_{m \in \Phi(n)} m = a^{\Phi(n)} \left(\prod_{m \in \Phi(n)} m \right) \pmod n.$$

- Finally, $a^{\Phi(n)} = 1 \pmod n$ because $n \nmid \prod_{m \in \Phi(n)} m$.

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,

$$5^4 = 625 = 1 \pmod{12}.$$