TRIPARTITE MATCHING\textsuperscript{a}

• 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 $H$.

\textbf{Theorem 49 (Karp, 1972)} TRIPARTITE MATCHING \textit{is NP-complete}.

\textsuperscript{a}Princess Diana (November 20, 1995), “There were three of us in this marriage, so it was a bit crowded.”
Related Problems

- We are given a family $F = \{S_1, S_2, \ldots, S_n\}$ 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 (X3C) asks if there are $m$ sets in $F$ that are disjoint (so have $U$ as their union).
SET COVERING  SET PACKING
Related Problems (concluded)

Corollary 50 (Karp, 1972) \textsc{set covering, set packing, and x3c} are all \textit{NP-complete}.

- Does \textsc{set covering} remain \textit{NP-complete} when $|S_i| = 3^a$

- \textsc{set covering} is used to prove that the influence maximization problem in social networks is \textit{NP-complete}.\(^b\)

\(^a\)Contributed by Mr. Kai-Yuan Hou (B99201038, R03922014) on September 22, 2015.

\(^b\)Kempe, Kleinberg, & Tardos (2003).
KNAPSACK

- 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
  $$I \subseteq \{1, 2, \ldots, n\}$$
  such that $\sum_{i \in I} w_i \leq W$ and $\sum_{i \in I} v_i \geq K$.
  - We want to achieve the maximum satisfaction within the budget.
KNAPSACK Is NP-Complete\textsuperscript{a}

- \textsc{Knapsack} $\in$ NP: Guess an $I$ and check the constraints.
- We shall reduce X3C to \textsc{Knapsack}, in which $v_i = w_i$ for all $i$ and $K = W$.
- The simplified \textsc{Knapsack} now asks if a subset of $v_1, v_2, \ldots, v_n$ adds up to exactly $K$\textsuperscript{b}.
  
  - Picture yourself as a radio DJ.

\textsuperscript{a}Karp (1972). It can be solved in time $O(2^{n/2})$ with space $O(2^{n/4})$ (Schroeppel & Shamir, 1981; Vyskoč, 1987).

\textsuperscript{b}This problem is called \textsc{subset sum} or 0-1 \textsc{Knapsack}.
The Proof (continued)

• The primary differences between the two problems are:\(^a\)
  – Sets vs. numbers.
  – Union vs. addition.

• We are given a family \( F = \{S_1, S_2, \ldots, S_n\} \) of size-3 subsets of \( U = \{1, 2, \ldots, 3m\} \).

• \textsc{x3c} asks if there are \( m \) disjoint sets in \( F \) that cover the set \( U \).

\(^a\)Thanks to a lively class discussion on November 16, 2010.
The Proof (continued)

- Think of a set as a bit vector in \( \{0, 1\}^{3m} \).
  - Assume \( m = 3 \).
  - 110010000 means the set \( \{1, 2, 5\} \).
  - 001100010 means the set \( \{3, 4, 8\} \).
- Assume there are \( n = 5 \) size-3 subsets in \( F \).
- Our goal is
  \[
  \{11 \cdots 1\}^{3m}
  \]
The Proof (continued)

- A bit vector can also be seen as a binary *number*.
- Set union resembles addition:

\[
\begin{array}{c}
001100010 \\
+ 110010000 \\
\hline
111110010
\end{array}
\]

which denotes the set \( \{ 1, 2, 3, 4, 5, 8 \} \), as desired.
The Proof (continued)

- Trouble occurs when there is *carry*:

\[
\begin{array}{c}
010000000 \\
+ \quad 010000000 \\
\hline
100000000
\end{array}
\]

which denotes the wrong set \{1\}, not the correct \{2\}. 
The Proof (continued)

- Or consider

\[
\begin{array}{c}
001100010 \\
+ 001110000 \\
\hline
011010010
\end{array}
\]

which denotes the set \{2, 3, 5, 8\}, not the correct \{3, 4, 5, 8\}\textsuperscript{a}.

\textsuperscript{a}Corrected by Mr. Chihwei Lin (D97922003) on January 21, 2010.
The Proof (continued)

- Carry may also lead to a situation where we obtain our solution $11 \cdots 1$ with more than $m$ sets in $F$.

- For example, with $m = 3$,

\[
\begin{align*}
000100010 \\
001110000 \\
101100000 \\
\end{align*}
\]

\[
\begin{array}{c}
+ \\
000001101 \\
111111111
\end{array}
\]

- But the correct union result, \{1, 3, 4, 5, 6, 7, 8, 9\}, is not an exact cover.
The Proof (continued)

- And it uses 4 sets instead of the required \( m = 3 \).\(^a\)

- To fix this problem, we enlarge the base just enough so that there are no carries.\(^b\)

- Because there are \( n \) vectors in total, we change the base from 2 to \( n + 1 \).

- Every positive integer \( N \) has a unique expression in base \( b \): There are \textbf{\( b \)-adic digits} \( 0 \leq d_i < b \) such that

\[
N = \sum_{i=0}^{k} d_i b^i, \quad d_k \neq 0.
\]

\(^a\)Thanks to a lively class discussion on November 20, 2002.

\(^b\)You cannot map \( \cup \) to \( \vee \) because \textsc{knapsack} requires \( + \) not \( \vee \)!
The Proof (continued)

• Set $v_i$ to be the integer corresponding to the bit vector encoding $S_i$ in base $n + 1$:

$$v_i = \sum_{j \in S_i} 1 \times (n + 1)^{3m-j}$$ (4)

• Set

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

• Now in base $n + 1$, if there is a set $S$ such that

$$\sum_{i \in S} v_i = \overline{11 \cdots 1},$$

then every position must be contributed by exactly one $v_i$ and $|S| = m$. 
The Proof (continued)

• For example, the case on p. 429 becomes

\[
\begin{align*}
000100010 \\
001110000101100000 \\
+ 000001101 \\
102311111
\end{align*}
\]

in base \(n + 1 = 6\).

• As desired, it no longer meets the goal.
The Proof (continued)

• Suppose $F$ admits an exact cover, say $\{S_1, S_2, \ldots, S_m\}$.

• Then picking $I = \{1, 2, \ldots, m\}$ clearly results in

\[
v_1 + v_2 + \cdots + v_m = 11\ldots1.
\]

• It is important to note that the meaning of addition ($+$) is independent of the base.\(^a\)

  – It is just regular addition.

  – But an $S_i$ may give rise to different integers $v_i$ in Eq. (4) on p. 431 under different bases.

\(^a\)Contributed by Mr. Kuan-Yu Chen (R92922047) on November 3, 2004.
The Proof (concluded)

• On the other hand, suppose there exists an $I$ such that

$$\sum_{i \in I} v_i = \overbrace{11 \cdots 1}^{3m}$$

in base $n + 1$.

• The no-carry property implies that $|I| = m$ and

$$\{ S_i : i \in I \}$$

is an exact cover.

The proof actually proves:

**Corollary 51** SUBSET SUM (p. 423) is NP-complete.
An Example

• Let \( m = 3 \), \( U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \), and

\[
\begin{align*}
S_1 &= \{1, 3, 4\}, \\
S_2 &= \{2, 3, 4\}, \\
S_3 &= \{2, 5, 6\}, \\
S_4 &= \{6, 7, 8\}, \\
S_5 &= \{7, 8, 9\}.
\end{align*}
\]

• Note that \( n = 5 \), as there are 5 \( S_i \)'s.
An Example (continued)

- Our reduction produces

\[
K = \sum_{j=0}^{3\times3-1} 6^j = 11\cdots16 = 2015539_{10},
\]

\[
v_1 = 101100000 = 1734048,
\]

\[
v_2 = 011100000 = 334368,
\]

\[
v_3 = 010011000 = 281448,
\]

\[
v_4 = 000001110 = 258,
\]

\[
v_5 = 000000111 = 43.
\]
An Example (concluded)

• Note \( v_1 + v_3 + v_5 = K \) because

\[
\begin{align*}
101100000 & \quad 010011000 \\
010011000 & \quad + \quad 000000111 \\
\hline
111111111 & \\
\end{align*}
\]

• 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 PACKING

• We are given $N$ positive integers $a_1, a_2, \ldots, a_N$, 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 52** BIN PACKING is NP-complete.
BIN PACKING (concluded)

• But suppose $a_1, a_2, \ldots, a_N$ are randomly distributed between 0 and 1.

• Let $B$ be the smallest number of unit-capacity bins capable of holding them.

• Then $B$ can deviate from its average by more than $t$ with probability at most $2e^{-2t^2/N}$.

---

\[a\text{Dubhashi & Panconesi (2012).}\]
INTEGER PROGRAMMING (IP)

• IP asks whether a system of linear inequalities with integer coefficients has an integer solution.

• In contrast, LINEAR PROGRAMMING (LP) asks whether a system of linear inequalities with integer coefficients has a *rational* solution.
  
  – LP is solvable in polynomial time.a

---

aKhachiyan (1979).
IP Is NP-Complete\textsuperscript{a}

- \textbf{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

  - $x_i = 1$ 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.
  
  - The $i$th row of $Ax$ is at least 1 means item $i$ is covered.

\textsuperscript{a}Karp (1972); Borosh & Treybig (1976); Papadimitriou (1981).
IP Is NP-Complete (concluded)

- This shows IP is NP-hard.
- Many NP-complete problems can be expressed as an IP problem.
- IP with a fixed number of variables is in $P$.\(^a\)

\(^a\)Lenstra (1983).
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. 365) and the **KNAPSACK** proof (p. 423): equally hard.
  - **CIRCUIT VALUE** to **MONOTONE CIRCUIT VALUE** (p. 314): equally hard.
  - **SAT** to **2SAT** (p. 346): 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. 398): harder.
  - LP to IP (p. 440): harder.
  - SAT to NAESAT (p. 358) and MAX CUT to MAX BISECTION (p. 396): equally hard.
  - 3-COLORING to 2-COLORING (p. 407): easier.
coNP and Function Problems
coNP

• By definition, coNP is the class of problems whose complement is in NP.
  – $L \in \text{coNP}$ if and only if $\overline{L} \in \text{NP}$.

• NP problems have succinct certificates.$^a$

• coNP is therefore the class of problems that have succinct disqualifications:$^b$
  – A “no” instance possesses a short proof of its being a “no” instance.
  – Only “no” instances have such proofs.

---

$^a$Recall Proposition 40 (p. 328).
$^b$To be proved in Proposition 53 (p. 456).
coNP (continued)

• Suppose $L$ is a coNP problem.

• There exists a nondeterministic polynomial-time algorithm $M$ such that:
  - If $x \in L$, then $M(x) = \text{“yes”}$ for all computation paths.
  - If $x \not\in L$, then $M(x) = \text{“no”}$ for some computation path.

• If we swap “yes” and “no” in $M$, the new algorithm decides $\overline{L} \in \text{NP}$ in the classic sense (p. 107).
$x \in L$

$\begin{array}{c}
\text{yes} \\
\text{yes} \\
\text{yes}
\end{array}$

$\begin{array}{c}
\text{yes} \\
\text{yes}
\end{array}$

$x \notin L$

$\begin{array}{c}
\text{no} \\
\text{yes} \\
\text{no}
\end{array}$

$\begin{array}{c}
\text{yes} \\
\text{yes}
\end{array}$
coNP (continued)

- So there are 3 major approaches to proving $L \in \text{coNP}$.
  1. Prove $\overline{L} \in \text{NP}$.
  2. Prove that only “no” instances possess short proofs.
  3. Write an algorithm for it directly.
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 p. 155).
Some coNP Problems

• SAT COMPLEMENT $\in$ coNP.
  – SAT COMPLEMENT is the complement of SAT.
  – Or, the disqualification is a truth assignment that satisfies it.

• HAMILTONIAN PATH COMPLEMENT $\in$ coNP.
  – HAMILTONIAN PATH COMPLEMENT is the complement of HAMILTONIAN PATH.
  – Or, the disqualification is a Hamiltonian path.
Some coNP Problems (concluded)

• **VALIDITY** \(\in\) coNP.
  
  – If \(\phi\) is not valid, it can be disqualified very succinctly: a truth assignment that does not satisfy it.

• **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. 117)

\( \phi \) is a boolean formula with \( n \) variables.

1: \textbf{for} \( i = 1, 2, \ldots, n \) \textbf{do}
2: \hspace{1em} Guess \( x_i \in \{0, 1\} \); \{Nondeterministic choice.\}
3: \textbf{end for}
4: \{Verification:\}
5: \textbf{if} \( \phi(x_1, x_2, \ldots, x_n) = 1 \) \textbf{then}
6: \hspace{1em} “no”;
7: \textbf{else}
8: \hspace{1em} “yes”;
9: \textbf{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 \( 2^n \).
  - \( \phi \) is unsatisfiable if and only if every truth assignment falsifies \( \phi \).
  - But every truth assignment falsifies \( \phi \) if and only if every computation path results in “yes.”
An Alternative Characterization of coNP

**Proposition 53** 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. 327, we assume $|y| \leq |x|^k$ for some $k$.)

- $\overline{L} = \{ x : \exists y (x, y) \in \neg R \}.$
- Because $\neg R$ remains polynomially balanced, $\overline{L} \in \text{NP}$ by Proposition 40 (p. 328).
- Hence $L \in \text{coNP}$ by definition.
coNP-Completeness

**Proposition 54**  
$L$ is NP-complete if and only if its complement $\overline{L} = \Sigma^* - L$ is coNP-complete.

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

- Let $\overline{L}'$ 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, $x \not\in L'$ if and only if $R(x) \not\in L$. 
coNP Completeness (concluded)

• So $x \in \overline{L}'$ if and only if $R(x) \in \overline{L}$.

• The same $R$ is a reduction from $\overline{L}'$ to $\overline{L}$.

• This shows $\overline{L}$ is coNP-hard.

• But $\overline{L} \in \text{coNP}$.

• This shows $\overline{L}$ is coNP-complete.
Some coNP-Complete Problems

• SAT COMPLEMENT is coNP-complete.

• HAMILTONIAN PATH COMPLEMENT is coNP-complete.

• VALIDITY is coNP-complete.
  – φ is valid if and only if ¬φ is not satisfiable.
  – φ ∈ VALIDITY is valid if and only if ¬φ ∈ SAT COMPLEMENT.
  – The reduction from SAT COMPLEMENT to VALIDITY is hence easy.
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 \textbf{prime} if \( p > 1 \) and all positive numbers other than 1 and \( p \) itself cannot divide it.

- \textsc{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 \textsc{primes} was not found until 2002 by Agrawal, Kayal, and Saxena!

- The running time is \( \tilde{O}(\log^{7.5} N) \).
1: if \( n = a^b \) for some \( a, b > 1 \) then
2: \hspace{0.5em} return “composite”;
3: end if
4: for \( r = 2, 3, \ldots, n - 1 \) do
5: \hspace{0.5em} if \( \gcd(n, r) > 1 \) then
6: \hspace{1em} return “composite”;
7: end if
8: if \( r \) is a prime then
9: \hspace{1em} Let \( q \) be the largest prime factor of \( r - 1 \);
10: \hspace{1em} if \( q \geq 4\sqrt{r} \log n \) and \( n^{(r-1)/q} \neq 1 \mod r \) then
11: \hspace{2em} 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 \( a = 1, 2, \ldots, 2\sqrt{r} \log n \) do
16: \hspace{1em} if \( (x - a)^n \neq (x^n - a) \mod (x^r - 1) \) in \( \mathbb{Z}_n[x] \) then
17: \hspace{2em} return “composite”;
18: end if
19: end for
20: 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.).

- NP ∩ 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 ∈ NP ∩ coNP.
  - In fact, PRIMES ∈ P as mentioned earlier.
Basic Modular Arithmetics\textsuperscript{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 $\text{gcd}(m, n)$. 

• The $r$ in Theorem 55 (p. 466) is a primitive root of $p$. 

\textsuperscript{a}Carl Friedrich Gauss.
Basic Modular Arithmetics (concluded)

- We use

\[ a \equiv b \mod n \]

if \( n \mid (a - b) \).
- So \( 25 \equiv 38 \mod 13 \).

- We use

\[ a = b \mod n \]

if \( b \) is the remainder of \( a \) divided by \( n \).
- So \( 25 = 12 \mod 13 \).
Primitive Roots in Finite Fields

Theorem 55 (Lucas & Lehmer, 1927) \(^a\) A number \(p > 1\) is a prime if and only if there is a number \(1 < r < p\) such that

1. \(r^{p-1} = 1 \mod p, \text{ and}\)

2. \(r^{(p-1)/q} \neq 1 \mod p\) for all prime divisors \(q\) of \(p - 1\).

- This \(r\) is called the **primitive root** or **generator**.
- We will prove one direction of the theorem later.\(^b\)

---

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

\(^b\)See pp. 477ff.
Derrick Lehmer\textsuperscript{a} (1905–1991)

\textsuperscript{a}Inventor of the linear congruential generator in 1951.
Pratt’s Theorem

Theorem 56 (Pratt, 1975) \( \text{PRIMES} \in NP \cap coNP \).

- \( \text{PRIMES} \in coNP \) because a succinct disqualification is a proper divisor.
  - A proper divisor of a number means it is not a prime.
- Now suppose \( p \) is a prime.
- \( p \)'s certificate includes the \( r \) in Theorem 55 (p. 466).
  - There may be multiple choices for \( r \).
The Proof (continued)

- Use recursive doubling to check if \( r^{p-1} = 1 \) mod \( p \) in time polynomial in the length of the input, \( \log_2 p \).
  - \( r, r^2, r^4, \ldots \) mod \( p \), a total of \( \sim \log_2 p \) steps.

- 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.

- Checking \( r^{(p-1)/q_i} \neq 1 \) mod \( p \) is also easy.

- Checking \( q_1, q_2, \ldots, q_k \) are all the divisors of \( p - 1 \) is easy.
The Proof (concluded)

- We still need certificates for the primality of the $q_i$’s.
- The complete certificate is recursive and tree-like:
  \[ C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)). \quad (5) \]
- 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 57  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 $q_1 = 2$,
  
  $$\prod_{i=2}^{k} q_i \leq \frac{p - 1}{2}.$$  \hspace{1cm} (6)
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,

$$|C(p)| \leq 5 \log_2 p + 5 + 5 \sum_{i=2}^{k} \log_2^2 q_i$$

$$\leq 5 \log_2 p + 5 + 5 \left( \sum_{i=2}^{k} \log_2 q_i \right)^2$$

$$\leq 5 \log_2 p + 5 + 5 \log_2^2 \frac{p - 1}{2} \quad \text{by inequality (6)}$$

$$< 5 \log_2 p + 5 + 5 \left( \log_2 p \right)^2$$

$$= 5 \log_2^2 p + 10 - 5 \log_2 p \leq 5 \log_2^2 p$$

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

• How to turn the proof into a nondeterministic polynomial-time algorithm?

• First, guess a $\log_2 p$-bit number $r$.

• Then guess up to $\log_2 p$ numbers $q_1, q_2, \ldots, q_k$ each containing at most $\log_2 p$ bits.

• Then recursively do the same thing for each of the $q_i$ to form a certificate (5) on p. 470.

• Finally check if the two conditions of Theorem 55 (p. 466) hold throughout the tree.

---

*aContributed by Mr. Kai-Yuan Hou (B99201038, R03922014) on November 24, 2015.*
Euler’s\textsuperscript{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 \( n \).\textsuperscript{b}
  \[- \Phi(12) = \{ 1, 5, 7, 11 \}.\]

- Define \textbf{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.

\textsuperscript{a}Leonhard Euler (1707–1783).
\textsuperscript{b}\( \mathbb{Z}_n^* \) is an alternative notation.
Leonhard Euler (1707–1783)
Three Properties of Euler’s Function\textsuperscript{a}

The inclusion-exclusion principle\textsuperscript{b} can be used to prove the following.

**Lemma 58** If $n = p_1^{e_1} p_2^{e_2} \cdots p_\ell^{e_\ell}$ is the prime factorization of $n$, then

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

- For example, if $n = pq$, where $p$ and $q$ are distinct primes, then

$$
\phi(n) = pq \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) = pq - p - q + 1.
$$

\textsuperscript{a}See p. 224 of the textbook.

\textsuperscript{b}Consult any textbooks on discrete mathematics.
Three Properties of Euler’s Function (concluded)

Corollary 59 $\phi(mn) = \phi(m) \phi(n)$ if $\text{gcd}(m, n) = 1$.

Lemma 60 $\sum_{m|n} \phi(m) = n$. 
The Density Attack for PRIMES

All numbers $< n$

Witnesses to compositeness of $n$
The Density Attack for PRIMES

1: Pick \( k \in \{1, \ldots, n\} \) randomly;
2: \textbf{if} \( k \mid n \) and \( k \neq 1 \) and \( k \neq n \) \textbf{then}
3: \hspace{1em} \textbf{return} “\( n \) is composite”;
4: \hspace{1em} \textbf{else}
5: \hspace{2em} \textbf{return} “\( n \) is (probably) a prime”;
6: \hspace{1em} \textbf{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 gray 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: \( \Omega(2^{0.5 \log_2 n}) \).