

# Number-Theoretic Algorithms

Hengfeng Wei

hfwei@nju.edu.cn

March 31 ~ April 4, 2017

# Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm
- 3 Primes
- 4 Chinese Remainder Theorem

# Cancellation in modular arithmetic

(TC 31.4.2)

$$ad \equiv bd \pmod{n} \not\Rightarrow a \equiv b \pmod{n}$$

# Cancellation in modular arithmetic

(TC 31.4.2)

$$ad \equiv bd \pmod{n} \not\Rightarrow a \equiv b \pmod{n}$$

$$3 \cdot 2 \equiv 5 \cdot 2 \pmod{4} \quad 3 \not\equiv 5 \pmod{4}$$

# Cancellation in modular arithmetic

(TC 31.4.2)

$$ad \equiv bd \pmod{n} \not\Rightarrow a \equiv b \pmod{n}$$

$$ad \equiv bd \pmod{n}, a \perp n \Rightarrow a \equiv b \pmod{n}$$

$$3 \cdot 2 \equiv 5 \cdot 2 \pmod{4} \quad 3 \not\equiv 5 \pmod{4}$$

# Changing the modulus

$$3 \cdot 2 \equiv 5 \cdot 2 \pmod{4} \quad 3 \not\equiv 5 \pmod{4} \quad 3 \equiv 5 \pmod{2}$$

# Changing the modulus

$$3 \cdot 2 \equiv 5 \cdot 2 \pmod{4} \quad 3 \not\equiv 5 \pmod{4} \quad 3 \equiv 5 \pmod{2}$$

$$ad \equiv bd \pmod{nd} \iff a \equiv b \pmod{n} \quad (d \neq 0)$$

# Changing the modulus

$$3 \cdot 2 \equiv 5 \cdot 2 \pmod{4} \quad 3 \not\equiv 5 \pmod{4} \quad 3 \equiv 5 \pmod{2}$$

$$ad \equiv bd \pmod{nd} \iff a \equiv b \pmod{n} \quad (d \neq 0)$$

$$ad \equiv bd \pmod{n} \iff a \equiv b \pmod{\frac{n}{(d, n)}}$$



# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n} \implies a \equiv b \pmod{n_i}$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n} \implies a \equiv b \pmod{n_i}$$

$$a \equiv b \pmod{100} \implies a \equiv b \pmod{20} \implies a \equiv b \pmod{5}$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{[n_1, n_2]}$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{[n_1, n_2]}$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{n_1 n_2}, \text{ if } n_1 \perp n_2$$

# Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{[n_1, n_2]}$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{n_1 n_2}, \text{ if } n_1 \perp n_2$$

$$\forall 1 \leq i \leq k, a \equiv b \pmod{n_i} \iff a \equiv b \pmod{n}, \text{ if } n_i \perp n_j$$

# Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm**
- 3 Primes
- 4 Chinese Remainder Theorem



# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

1. If  $a > b \geq 0$ ,  $\text{EUCLID}(a, b)$  makes  $\leq r \triangleq 1 + \log_{\phi} b$  recursive calls.

$$a > b \geq 1, b < F_{k+1} \implies r < k.$$

$$r \leq 1 + \log_{\phi} b \implies k = 2 + \log_{\phi} b, b < F_{3+\log_{\phi} b}$$

$$F_k = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}} = \left\lfloor \frac{\phi^k}{\sqrt{5}} \right\rfloor \geq \frac{\phi^k}{\sqrt{5}}$$

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

1. If  $a > b \geq 0$ ,  $\text{EUCLID}(a, b)$  makes  $\leq r \triangleq 1 + \log_{\phi} b$  recursive calls.

$$a > b \geq 1, b < F_{k+1} \implies r < k.$$

$$r \leq 1 + \log_{\phi} b \implies k = 2 + \log_{\phi} b, b < \boxed{?} \leq F_{3+\log_{\phi} b}$$

$$F_k = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}} = \left\lfloor \frac{\phi^k}{\sqrt{5}} \right\rfloor \geq \frac{\phi^k}{\sqrt{5}}$$

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_{\phi}\left(\frac{b}{(a,b)}\right)$ .

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_{\phi}(\frac{b}{(a,b)})$ .

$$(a, b) = (a, b) \cdot (\frac{a}{(a, b)}, \frac{b}{(a, b)})$$

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_\phi(\frac{b}{(a,b)})$ .

$$(a, b) = (a, b) \cdot (\frac{a}{(a, b)}, \frac{b}{(a, b)})$$

$$(16, 12)$$

$$= (12, 4)$$

$$= (4, 0)$$

$$= 4$$

$$(4, 3)$$

$$= (3, 1)$$

$$= (1, 0)$$

$$= 1$$

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_\phi(\frac{b}{(a,b)})$ .

$$(a, b) = (a, b) \cdot \left( \frac{a}{(a, b)}, \frac{b}{(a, b)} \right)$$

$(16, 12)$	$(4, 3)$
$= (12, 4)$	$= (3, 1)$
$= (4, 0)$	$= (1, 0)$
$= 4$	$= 1$
$\text{EUCLID}(a, b) \leftrightarrow \text{EUCLID}\left(\frac{a}{(a, b)}, \frac{b}{(a, b)}\right)$	

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_\phi(\frac{b}{(a,b)})$ .

$$\text{EUCLID}(a, b) \leftrightarrow \text{EUCLID}\left(\frac{a}{(a, b)}, \frac{b}{(a, b)}\right)$$

$$\text{EUCLID}(b, a \bmod b) \leftrightarrow \text{EUCLID}\left(\frac{b}{(a, b)}, \frac{a}{(a, b)} \bmod \frac{b}{(a, b)}\right)$$

$$\frac{a}{(a, b)} \bmod \frac{b}{(a, b)} = \frac{a \bmod b}{(a, b)}$$

# Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

2. Improve this bound to  $1 + \log_{\phi}\left(\frac{b}{(a,b)}\right)$ .

Lemma (Generalization of Lemma 31.10)

*If  $a > b \geq 1$ ,  $d = (a, b)$  and  $\text{EUCLID}(a, b)$  performs  $k \geq 1$  recursive calls, then  $a \geq dF_{k+2}$  and  $b \geq dF_{k+1}$ .*



# Average-case analysis of Euclid's algorithm

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n) \quad n \geq 1$$

# Average-case analysis of Euclid's algorithm

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n) \quad n \geq 1$$

When  $m$  is chosen at random:

$$T_n = \frac{1}{n} \sum_{0 \leq k < n} T(k, n)$$

# Average-case analysis of Euclid's algorithm

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n) \quad n \geq 1$$

When  $m$  is chosen at random:

$$T_n = \frac{1}{n} \sum_{0 \leq k < n} T(k, n)$$

Assume that, for  $0 \leq k < n$ ,  $(n \bmod k)$  is “random”:

$$T_n \approx 1 + \frac{1}{n}(T_0 + T_1 + \cdots + T_{n-1})$$

# Average-case analysis of Euclid's algorithm

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n) \quad n \geq 1$$

When  $m$  is chosen at random:

$$T_n = \frac{1}{n} \sum_{0 \leq k < n} T(k, n)$$

Assume that, for  $0 \leq k < n$ ,  $(n \bmod k)$  is “random”:

$$T_n \approx 1 + \frac{1}{n}(T_0 + T_1 + \cdots + T_{n-1}) = 1 + \frac{1}{2} + \cdots + \frac{1}{n} = H_n \approx \ln n + O(1)$$

# Average-case analysis of Euclid's algorithm

$$T(m, 0) = 0; \quad T(m, n) = 1 + T(n, m \bmod n) \quad n \geq 1$$

When  $m$  is chosen at random:

$$T_n = \frac{1}{n} \sum_{0 \leq k < n} T(k, n)$$

Assume that, for  $0 \leq k < n$ ,  $(n \bmod k)$  is “random”:

$$T_n \approx 1 + \frac{1}{n}(T_0 + T_1 + \cdots + T_{n-1}) = 1 + \frac{1}{2} + \cdots + \frac{1}{n} = H_n \approx \ln n + O(1)$$

## Reference

“The Art of Computer Programming, Vol 2: Seminumerical Algorithms (Section 4.5.3)” by Donald E. Knuth, 3rd edition.

# Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm
- 3 Primes**
- 4 Chinese Remainder Theorem

# Pairwise relatively prime

(TC 31.2–9)

$n_1, n_2, n_3, n_4$  are pairwise relatively prime

$\iff$

$$\gcd(n_1n_2, n_3n_4) = \gcd(n_1n_3, n_2n_4) = 1$$

# Pairwise relatively prime

(TC 31.2–9)

$n_1, n_2, \dots, n_k$  are pairwise relatively prime



a set of  $\lceil \lg k \rceil$  pairs of numbers derived from the  $n_i$  are relatively prime.



# Pairwise relatively prime

(TC 31.2–9)

$n_1, n_2, \dots, n_k$  are pairwise relatively prime



a set of  $\lceil \lg k \rceil$  pairs of numbers derived from the  $n_i$  are relatively prime.

$$\binom{k}{2} = \Theta(k^2) \quad (\text{complete graph})$$

# Pairwise relatively prime

(TC 31.2–9)

$n_1, n_2, \dots, n_k$  are pairwise relatively prime



a set of  $\lceil \lg k \rceil$  pairs of numbers derived from the  $n_i$  are relatively prime.

$$\binom{k}{2} = \Theta(k^2) \quad (\text{complete graph})$$

$$\gcd(\boxed{1_L}, \boxed{1_R}) = \gcd(\boxed{2_L}, \boxed{2_R}) = \dots = \gcd(\boxed{\lceil \lg k \rceil_L}, \boxed{\lceil \lg k \rceil_R}) = 1$$

# Pairwise relatively prime

(TC 31.2–9)

$n_1, n_2, \dots, n_k$  are pairwise relatively prime



a set of  $\lceil \lg k \rceil$  pairs of numbers derived from the  $n_i$  are relatively prime.

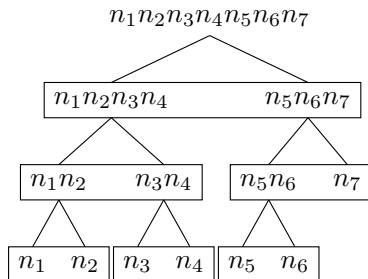
$$\binom{k}{2} = \Theta(k^2) \quad (\text{complete graph})$$

$$\gcd(\boxed{1_L}, \boxed{1_R}) = \gcd(\boxed{2_L}, \boxed{2_R}) = \dots = \gcd(\boxed{\lceil \lg k \rceil_L}, \boxed{\lceil \lg k \rceil_R}) = 1$$

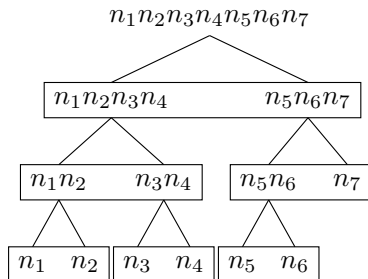
$$k = 3 : \quad \gcd(n_1, n_2 n_3) = \gcd(n_2, n_3) = 1$$

$$k = 2 : \quad \gcd(n_1, n_2) = 1$$

# Pairwise relatively prime: divide-and-conquer

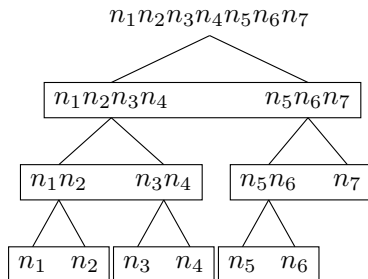


# Pairwise relatively prime: divide-and-conquer



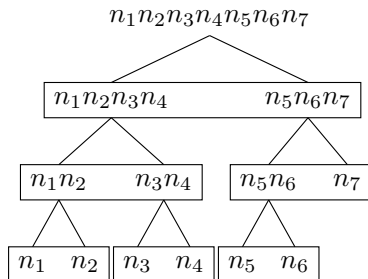
$$\begin{cases} T(1) = 0 \\ T(k) = 2T(\frac{k}{2}) + 1 \end{cases}$$

# Pairwise relatively prime: divide-and-conquer



$$\begin{cases} T(1) = 0 \\ T(k) = 2T(\frac{k}{2}) + 1 \end{cases} \implies T(k) = k - 1$$

# Pairwise relatively prime: divide-and-conquer



$$\begin{cases} T(1) = 0 \\ T(k) = 2T(\frac{k}{2}) + 1 \end{cases} \implies T(k) = k - 1$$

$$T_k = k - 1 : (n_i, n_{i+1}n_{i+2} \cdots n_k) \quad \forall 1 \leq i < k$$

# Pairwise relatively prime: smarter combination

TODO: figure here.

$$\begin{cases} T(1) = 0 \\ T(k) = T(\frac{k}{2}) + 1 \end{cases}$$



# Pairwise relatively prime: smarter combination

TODO: figure here.

$$\begin{cases} T(1) = 0 \\ T(k) = T(\frac{k}{2}) + 1 \end{cases} \implies T(k) = \lceil \lg k \rceil$$

# Pairwise relatively prime: the dividing pattern

$$n_0, n_1, n_2, \dots, n_{k-1}$$

# Can we do even better?

$$T(k) \geq \lceil \lg k \rceil.$$

# Can we do even better?

$$T(k) \geq \lceil \lg k \rceil.$$

Prove by (strong) mathematical induction.

# Can we do even better?

$$T(k) \geq \lceil \lg k \rceil.$$

Prove by (strong) mathematical induction.

$$\begin{aligned} T(k) &\geq 1 + T(\lceil \frac{k}{2} \rceil) \\ &\geq 1 + \lceil \lg \lceil \frac{k}{2} \rceil \rceil \\ &= \lceil \lg k \rceil \end{aligned}$$

# Biclique covering

Covering a complete graph with few complete bipartite subgraphs.

covering a graph by complete bipartite graphs

All Images Videos News More Settings Tools

About 780,000 results (0.48 seconds)

**Covering a graph by complete bipartite graphs - ScienceDirect**  
[www.sciencedirect.com/science/article/pii/S0012365X96001240](http://www.sciencedirect.com/science/article/pii/S0012365X96001240) ▼  
 by P Erdős · 1997 · Cited by 25 · Related articles  
 Jun 10, 1997 · We prove the following theorem: the edge set of every graph  $G$  on  $n$  vertices can be partitioned into the disjoint union of complete bipartite ...

**On covering graphs by complete bipartite subgraphs - Science Direct**  
[www.sciencedirect.com/science/article/pii/S0012365X08005566](http://www.sciencedirect.com/science/article/pii/S0012365X08005566) ▼  
 by S Jukna · 2009 · Cited by 18 · Related articles  
 We prove that, if a graph with  $n$  vertices contains  $m$  vertex-disjoint edges, then  $m/2 \log n$  complete bipartite subgraphs are necessary to cover all its edges. ... For sparse graphs, this improves the well-known tooling set lower bound in communication complexity. ... The biclique covering ...

**PDF Covering a graph by complete bipartite graphs - URI Math**  
[www.math.uri.edu/~eaton/Mia1.pdf](http://www.math.uri.edu/~eaton/Mia1.pdf) ▼  
 by P Erdős · Cited by 25 · Related articles  
 Covering a graph by complete bipartite graphs. P. Erdős, L. Pyber\*, Mathematical Institute of the Hungarian Academy of Sciences, P.O. Box 127, 1-1-1364 ...

**PDF On covering graphs by complete bipartite subgraphs**  
[lovelace.thi.informatik.uni-frankfurt.de/~jukna/tpl/covering.pdf](http://lovelace.thi.informatik.uni-frankfurt.de/~jukna/tpl/covering.pdf) ▼  
 by S Jukna · Cited by 18 · Related articles  
 edges of the graph  $G$  itself can be covered by  $O(2 \log n)$  complete subgraphs. ... relation between bipartite  $n \times n$  graphs with  $n = 2k$  and boolean functions is ...

**Covering a graph by complete bipartite graphs - ACM Digital Library**  
[dl.acm.org/citation.cfm?id=2781997](http://dl.acm.org/citation.cfm?id=2781997)  
 by P Erdős · 1997 · Cited by 25 · Related articles  
 Jun 10, 1997 · We prove the following theorem: the edge set of every graph  $G$  on  $n$  vertices can be partitioned into the disjoint union of complete bipartite ...

**PDF Covering Graphs with Few Complete Bipartite Subgraphs \***  
<https://www.ac.tuwien.ac.at/files/pub/FleischnerMujuniPaulusmaSzeider09.pdf> ▼  
 by H Fleischner · Cited by 9 · Related articles  
 Abstract. We consider computational problems on covering graphs with bicliques (complete bipartite subgraphs). Given a graph and an integer  $k$ , the biclique ...

# Biclique covering: rethinking the first divide-and-conquer

$$T(k) = k - 1$$

# Biclique covering: rethinking the first divide-and-conquer

$$T(k) = k - 1$$

*edge-disjoint* biclique partition



# Biclique covering: rethinking the first divide-and-conquer

$$T(k) = k - 1$$

*edge-disjoint* biclique partition

Reference for  $T(k) \geq k - 1$

“On the Addressing Problem for Loop Switching” by Graham and Pollak, 1971.

# Biclique covering: rethinking the first divide-and-conquer

$$T(k) = k - 1$$

*edge-disjoint* biclique partition

Reference for  $T(k) \geq k - 1$

“On the Addressing Problem for Loop Switching” by Graham and Pollak, 1971.

Reference for *weighted* biclique partition

“Covering a Graph by Complete Bipartite Graphs” by P. Erdos and L. Pyber, 1997.

# Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm
- 3 Primes
- 4 Chinese Remainder Theorem

# Chinese Remainder Theorem (CRT)

## Theorem (CRT)

$$n_1, \dots, n_k; \quad a_1, \dots, a_k$$

$$n_i \perp n_j \quad i \neq j, \quad n = n_1 n_2 \cdots n_k$$

$$\exists! a \ (0 \leq a < n) : a \equiv a_i \pmod{n_i}.$$

## Proof for uniqueness.

$$a \equiv a' \pmod{n_i} \implies n \mid a - a'.$$



# History of CRT

# Proof of CRT (1)

Nonconstructive proof.

$$f : [0, n) \rightarrow \prod_{1 \leq i \leq k} [0, a_i)$$

$$f : a \mapsto (a \pmod{n_1}, \dots, a \pmod{n_k})$$

- ▶  $f$  is one-to-one.
- ▶  $f$  is onto.

$$\exists a : f(a) = (a_1, \dots, a_k).$$



# Proof of CRT (2)

Constructive proof by induction.

$$a \equiv a_1 \pmod{n_1}$$

$$a \equiv a_2 \pmod{n_2}$$

$$n_1 \perp n_2 \implies n_1 n'_1 + n_2 n'_2 = 1$$

$$\begin{aligned} x &= a_1 n_1 n'_1 + a_2 n_2 n'_2 \pmod{n_1 n_2} \\ &= a_1 M_1 (M_1^{-1} \pmod{n_1}) \\ &\quad + a_2 M_2 (M_2^{-1} \pmod{n_2}) \pmod{n_1 n_2} \end{aligned}$$



# Proof of CRT (2)

$$a \equiv a_1 \pmod{n_1}$$

$$a \equiv a_2 \pmod{n_2}$$

Constructive proof by induction.

$$a = a_1 + n_1 y$$

$$n_1 y \equiv a_2 - a_1 \pmod{n_2}$$

$$y \equiv M_2^{-1}(a_2 - a_1) \pmod{n_2}$$

$$n_1 y \equiv M_2 M_2^{-1}(a_2 - a_1) \pmod{n_1 n_2}$$

$$x \equiv a_1 + M_2 M_2^{-1}(a_2 - a_1) \pmod{n_1 n_2}$$

$$\equiv a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \pmod{n_1 n_2}$$





# Proof of CRT (3)

## Constructive proof.

$$1. \quad x \equiv 1 \pmod{n_i}, \quad x \equiv 0 \pmod{n_j} \quad (i \neq j)$$

$$x = M_i(M_i^{-1} \pmod{n_i}) \implies x \equiv M_i M_i^{-1} \pmod{m}$$

$$2. \quad x \equiv a_i \pmod{n_i}, \quad x \equiv 0 \pmod{n_j} \quad (i \neq j)$$

$$x \equiv a_i M_i M_i^{-1} \pmod{m}$$

$$3. \quad a \equiv a_i \pmod{n_i}$$

$$a \equiv \sum_{1 \leq i \leq k} a_i M_i M_i^{-1} \pmod{m}$$



# Proof of CRT (3)

More efficient constructive proof.

## Reference

“The Residue Number System” by Garner, 1959.



## CRT

Meaning of Figure 31.3  
 $\equiv 1$  and  $\equiv 0$  elsewhere

## CRT

$$a \leftrightarrow (a_1, a_2, \dots, a_n)$$

$$a \pm b \leftrightarrow (a_1 \pm b_1, a_2 \pm b_2, \dots, a_n \pm b_n)$$

$$a \times b \leftrightarrow (a_1 \times b_1, a_2 \times b_2, \dots, a_n \times b_n)$$

## TC 31.5–3

$$a \leftrightarrow (a_1, a_2, \dots, a_n), (a, n) = 1 \implies a^{-1} \leftrightarrow (a_1^{-1}, a_2^{-1}, \dots, a_n^{-1})$$

Proof.

$$a^{-1} \equiv a_i^{-1} \pmod{n_i}$$

## CRT

$$a \leftrightarrow (a_1, a_2, \dots, a_n)$$

$$a \pm b \leftrightarrow (a_1 \pm b_1, a_2 \pm b_2, \dots, a_n \pm b_n)$$

$$a \times b \leftrightarrow (a_1 \times b_1, a_2 \times b_2, \dots, a_n \times b_n)$$

## TC 31.5–3

$$a \leftrightarrow (a_1, a_2, \dots, a_n), (a, n) = 1 \implies a^{-1} \leftrightarrow (a_1^{-1}, a_2^{-2}, \dots, a_n^{-1})$$

Proof.

$$a^{-1} \equiv a_i^{-1} \pmod{n_i} \iff \begin{cases} a \equiv a_i \pmod{n_i} \\ (a, n) = 1 \end{cases}$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\varphi(p) = p - 1$$

$$\varphi(p^k) = p^k - p^{k-1}$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$



# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$

$$\varphi(n) = \prod_{i=1}^r \varphi(p_i^{k_i})$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$

$$\varphi(n) = \prod_{i=1}^r \varphi(p_i^{k_i}) = \prod_{i=1}^r (p_i^{k_i} - p_i^{k_i-1})$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$

$$\varphi(n) = \prod_{i=1}^r \varphi(p_i^{k_i}) = \prod_{i=1}^r (p_i^{k_i} - p_i^{k_i-1}) = \prod_{i=1}^r p_i^{k_i} \left(1 - \frac{1}{p_i}\right)$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$\begin{aligned}\varphi(p) &= p - 1 \\ \varphi(p^k) &= p^k - p^{k-1}\end{aligned}$$

$$n = \prod_{i=1}^r p_i^{k_i}$$

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$

$$\varphi(n) = \prod_{i=1}^r \varphi(p_i^{k_i}) = \prod_{i=1}^r (p_i^{k_i} - p_i^{k_i-1}) = \prod_{i=1}^r p_i^{k_i} \left(1 - \frac{1}{p_i}\right) = n \prod_{i=1}^r \left(1 - \frac{1}{p_i}\right)$$

# The $\varphi$ function

## Theorem (The $\varphi$ function)

$$m \perp n \implies \varphi(mn) = \varphi(m)\varphi(n)$$

Proof.

$$U_m = \{a \bmod m, (a, m) = 1\}, U_n = \{a \bmod n, (a, n) = 1\},$$

$$U_{mn} = \{c \bmod mn, (c, mn) = 1\}$$

$$f : U_{mn} \rightarrow U_m \times U_n$$

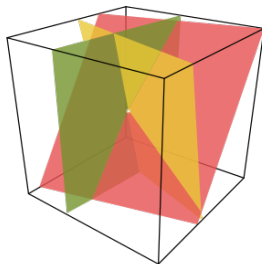
$$f(c \bmod mn) = (c \bmod m, c \bmod n).$$



# Secret sharing using the CRT

Definition ( $((k, n)$ -threshold secret sharing scheme)

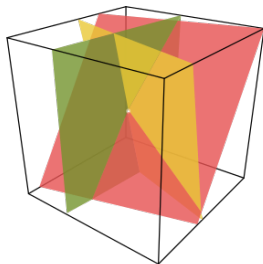
$(2, 3)$ -secret sharing:



# Secret sharing using the CRT

Definition ( $(k, n)$ -threshold secret sharing scheme)

$(2, 3)$ -secret sharing:



## Reference

“How to Share a Secret” by Mignotte, 1982.

# Secret sharing using the CRT

1. Choose  $m_i$ :

$$m_1 < m_2 < \cdots < m_n, \quad m_i \perp m_j, \quad \prod_{i=n-k+2}^n m_i < \prod_{i=1}^k m_i$$



# Secret sharing using the CRT

1. Choose  $m_i$ :

$$m_1 < m_2 < \cdots < m_n, \quad m_i \perp m_j, \quad \prod_{i=n-k+2}^n m_i < \prod_{i=1}^k m_i$$

2. Choose the secret  $S$ :

$$\prod_{i=n-k+2}^n m_i < S < \prod_{i=1}^k m_i$$

# Secret sharing using the CRT

1. Choose  $m_i$ :

$$m_1 < m_2 < \cdots < m_n, \quad m_i \perp m_j, \quad \prod_{i=n-k+2}^n m_i < \prod_{i=1}^k m_i$$

2. Choose the secret  $S$ :

$$\prod_{i=n-k+2}^n m_i < S < \prod_{i=1}^k m_i$$

3. Compute the shares:

$$s_i = S \bmod m_i$$

# Solving the system of congruences

(TC 31.5–2)

$$\begin{cases} x \equiv 1 \pmod{9} \\ x \equiv 2 \pmod{8} \\ x \equiv 3 \pmod{7} \end{cases}$$

# Solving the system of congruences

$$19x \equiv 556 \pmod{1155}$$

# Solving the system of congruences

## CRT with non-pairwise coprime moduli

$$\begin{cases} x \equiv 3 \pmod{8} \\ x \equiv 11 \pmod{20} \\ x \equiv 1 \pmod{15} \end{cases}$$