

Number-Theoretic Algorithms

Hengfeng Wei

hfwei@nju.edu.cn

March 31 ~ April 6, 2017



Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm
- 3 Pairwise Relatively Prime
- 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)

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

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

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

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

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

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

$$(a \bmod n)d = ad \bmod nd \quad (\text{distributive law})$$

Changing the modulus

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

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

$$(a \bmod n)d = ad \bmod nd \quad (\text{distributive law})$$

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

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

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{\text{lcm}(n_1, n_2)}$$

Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{\text{lcm}(n_1, n_2)}$$

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

Changing the modulus

$$n = n_1 n_2 \cdots n_k$$

$$a \equiv b \pmod{n_1}, a \equiv b \pmod{n_2} \iff a \equiv b \pmod{\text{lcm}(n_1, n_2)}$$

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

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

Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm**
- 3 Pairwise Relatively Prime
- 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 $\leq 1 + \log_{\phi} b$ recursive calls.

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

Lamé's theorem: $a > b \geq 1, b < F_{k+1} \implies r < k$.

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

Lamé's theorem: $a > b \geq 1, b < F_{k+1} \implies r < k$.

$$k = 2 + \log_{\phi} b$$

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

Lamé's theorem: $a > b \geq 1, b < F_{k+1} \implies r < k$.

$$k = 2 + \log_{\phi} b$$

To prove $b < F_{3+\log_{\phi} b}$.

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

Lamé's theorem: $a > b \geq 1, b < F_{k+1} \implies r < k$.

$$k = 2 + \log_{\phi} b$$

To prove $b < F_{3+\log_{\phi} b}$.

$$F_k = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}} > \frac{\phi^k - 1}{\sqrt{5}}$$

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 $1 + \log_{\phi}(\frac{b}{(a,b)})$.

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

$$\begin{aligned} (16, 12) \\ &= (12, 4) \\ &= (4, 0) \\ &= 4 \end{aligned}$$

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 $1 + \log_{\phi}\left(\frac{b}{(a,b)}\right)$.

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

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

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

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

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

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

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

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

Worst-case analysis of Euclid's algorithm

(TC 31.2–5)

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

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

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

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

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

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 Pairwise Relatively Prime**
- 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.

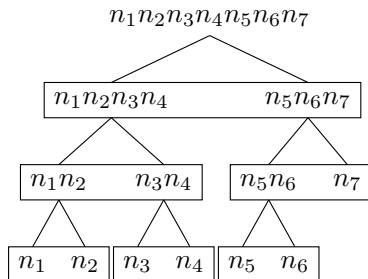
$$\binom{k}{2} = \Theta(k^2) \quad (\text{complete graph})$$

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

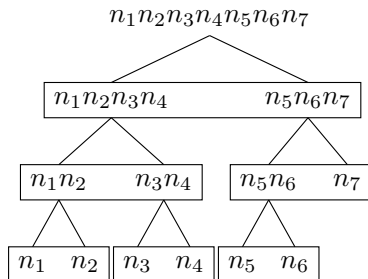
$$k = 2 : \quad \gcd(n_1, n_2) = 1$$

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

Pairwise relatively prime: divide-and-conquer

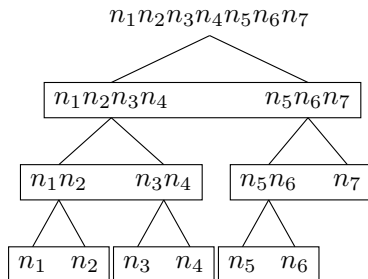


Pairwise relatively prime: divide-and-conquer



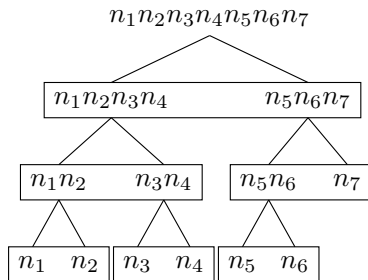
$$\begin{cases} T(1) = 0 \\ T(k) = T(\lceil \frac{k}{2} \rceil) + T(\lfloor \frac{k}{2} \rfloor) + 1 \end{cases}$$

Pairwise relatively prime: divide-and-conquer



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

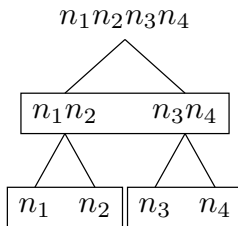
Pairwise relatively prime: divide-and-conquer



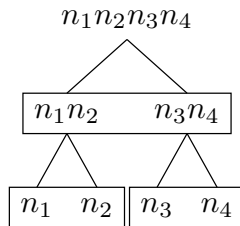
$$\begin{cases} T(1) = 0 \\ T(k) = T(\lceil \frac{k}{2} \rceil) + T(\lfloor \frac{k}{2} \rfloor) + 1 \end{cases} \implies T(k) = k - 1 = \Theta(k)$$

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

Pairwise relatively prime: smarter combination



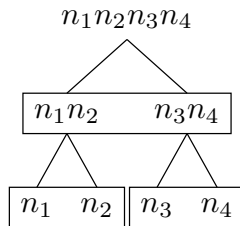
Pairwise relatively prime: smarter combination



$$(n_1n_2, n_3n_4) = 1$$

$$(n_1, n_2) = 1, (n_3, n_4) = 1$$

Pairwise relatively prime: smarter combination



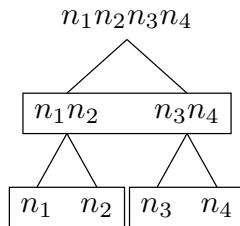
$$(n_1n_2, n_3n_4) = 1$$

$$(n_1, n_2) = 1, (n_3, n_4) = 1$$

$$(n_1n_2, n_3n_4) = 1$$

$$(n_1n_3, n_2n_4) = 1$$

Pairwise relatively prime: smarter combination



$$(n_1n_2, n_3n_4) = 1$$

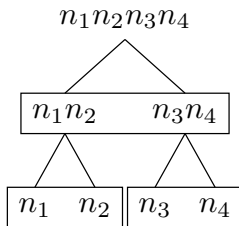
$$(n_1, n_2) = 1, (n_3, n_4) = 1$$

$$(n_1n_2, n_3n_4) = 1$$

$$(n_1n_3, n_2n_4) = 1$$

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

Pairwise relatively prime: smarter combination



$$(n_1n_2, n_3n_4) = 1$$

$$(n_1n_2, n_3n_4) = 1$$

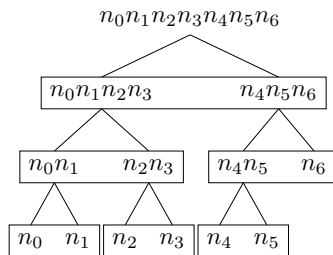
$$(n_1, n_2) = 1, (n_3, n_4) = 1$$

$$(n_1n_3, n_2n_4) = 1$$

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

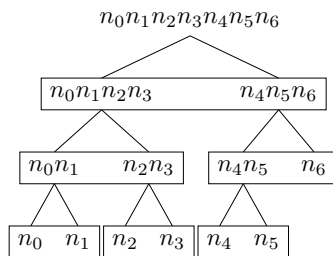
Pairwise relatively prime: the dividing pattern

$$k = 7 : \quad n_0, n_1, n_2, \dots, n_6$$



Pairwise relatively prime: the dividing pattern

$$k = 7 : \quad n_0, n_1, n_2, \dots, n_6$$



0 : 000

1 : 001

2 : 010

3 : 011

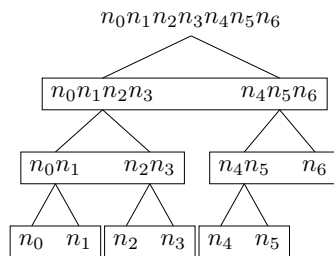
4 : 100

5 : 101

6 : 110

Pairwise relatively prime: the dividing pattern

$$k = 7 : \quad n_0, n_1, n_2, \dots, n_6$$



0 : 000

1 : 001

2 : 010

3 : 011

4 : 100

5 : 101

6 : 110

$$T(k) = \lceil \lg k \rceil$$

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 ▼
 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 ▼
 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 ▼
 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 ▼
 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
 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. Erdős and L. Pyber, 1997.

Number-Theoretic Algorithms

- 1 Modular Arithmetic
- 2 Euclid's Algorithm
- 3 Pairwise Relatively Prime
- 4 Chinese Remainder Theorem

Chinese Remainder Theorem (CRT)

Theorem (CRT)

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

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

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

Chinese Remainder Theorem (CRT)

Theorem (CRT)

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

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

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

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

Chinese Remainder Theorem (CRT)

Theorem (CRT)

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

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

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

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

Proof for uniqueness.

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



History of CRT

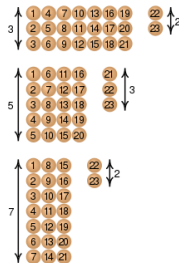
道題呢，說易是十分容易，說難卻又難到了極處。『今有物不知其數，三三數之賸二，五五數之賸三，七七數之賸二，問物幾何？』我知道這是二十三，不過那是硬湊出來的，要列一個每數皆可通用的算式，卻想破了腦袋也想不出。」

黃蓉笑道：「這容易得緊。以三三數之，餘數乘以七十；五五數之，餘數乘以二十一；七七數之，餘數乘十五。三者相加，如不大於一百零五，即為答數；否則須減去一百零五或其倍數。」瑛姑在心中盤算了一遍，果然絲毫不錯，低聲記誦道：「三三數之，餘數乘以七十；五五數之……」黃蓉道：「也不用這般硬記，我唸一首詩給你聽，那就容易記了：三人同行七十稀，五樹梅花廿一枝，七子團圓正半月，餘百零五便得知。」

History of CRT

道題呢，說易是十分容易，說難卻又難到了極處。『今有物不知其數，三三數之賸二，五五數之賸三，七七數之賸二，問物幾何？』我知道這是二十三，不過那是硬湊出來的，要列一個每數皆可通用的算式，卻想破了腦袋也想不出。」

黃蓉笑道：「這容易得緊。以三三數之，餘數乘以七十；五五數之，餘數乘以十五；七七數之，餘數乘以十五。三者相加，如不大於一百零五，即為答數；否則須減去一百零五或其倍數。」瑛姑在心中盤算了一遍，果然絲毫不錯，低聲記誦道：「三三數之，餘數乘以七十；五五數之……」黃蓉道：「也不用這般硬記，我唸一首詩給你聽，那就容易記了：三人同行七十稀，五樹梅花廿一枝，七子團圓正半月，餘百零五便得知。」



“物不知数”

History of CRT

今有物不知其數三三數之賸二五五數之賸三七七數之賸二問物幾何答曰二十三

術曰三三數之賸二置一百四十五五數之賸三置六十三七上數之賸二置三十并之得二百三十三以二百一十減之即得凡三三數之賸一則置七十五五數之賸一則置二十一七上數之賸一則置十五一百六以上以一百五減之即得

“孙子算经”

大衍總數術曰置諸問數類名有四一曰元數謂見位見單二曰收數謂見位見分度者數合冬三曰通數謂諸數各有分子母四曰和數謂諸數各有分子母

大衍數術

同而法則無二也然其極和較之用窮奇偶之情則有為元法西法所未及者但原本法解煩雜圖式謬舛今詳加改定並釋其義俾學者易見焉

秦九韶“数书九章”
大衍求一术

Proof of CRT (1)

Nonconstructive proof.

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

$$f : a \mapsto (a \bmod n_1, \dots, a \bmod n_k)$$

Proof of CRT (1)

Nonconstructive proof.

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

$$f : a \mapsto (a \bmod n_1, \dots, a \bmod n_k)$$

► f is one-to-one.



Proof of CRT (1)

Nonconstructive proof.

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

$$f : a \mapsto (a \bmod n_1, \dots, a \bmod n_k)$$

- ▶ 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.

$$a \equiv a_1 \pmod{n_1} \tag{1}$$

$$a \equiv a_2 \pmod{n_2} \tag{2}$$

Proof of CRT (2)

Constructive proof by induction.

$$a \equiv a_1 \pmod{n_1} \tag{1}$$

$$a \equiv a_2 \pmod{n_2} \tag{2}$$

$$(1) \implies a = a_1 + n_1 y$$

Proof of CRT (2)

Constructive proof by induction.

$$a \equiv a_1 \pmod{n_1} \tag{1}$$

$$a \equiv a_2 \pmod{n_2} \tag{2}$$

$$(1) \implies a = a_1 + n_1 y$$

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



Proof of CRT (3)

Constructive proof by induction.

$$a \equiv a_1 \pmod{n_1} \tag{3}$$

$$a \equiv a_2 \pmod{n_2} \tag{4}$$

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

Proof of CRT (3)

Constructive proof by induction.

$$a \equiv a_1 \pmod{n_1} \quad (3)$$

$$a \equiv a_2 \pmod{n_2} \quad (4)$$

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

$$x = a_1 n_1 n'_1 + a_2 n_2 n'_2 \pmod{n_1 n_2}$$



Proof of CRT (4)

Constructive proof.

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

$$x = M_i(M_i^{-1} \bmod n_i) \implies x = M_i M_i^{-1} \pmod{n}$$



Proof of CRT (4)

Constructive proof.

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

$$x = M_i(M_i^{-1} \bmod n_i) \implies x = M_i M_i^{-1} \pmod{n}$$

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

$$x = a_i M_i M_i^{-1} \pmod{n}$$



Proof of CRT (4)

Constructive proof.

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

$$x = M_i(M_i^{-1} \bmod n_i) \implies x = M_i M_i^{-1} \pmod{n}$$

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

$$x = a_i M_i M_i^{-1} \pmod{n}$$

$$3. \ a \equiv a_i \pmod{n_i}, \forall 1 \leq i \leq k$$

$$a = \sum_{1 \leq i \leq k} a_i M_i M_i^{-1} \pmod{n}$$



Proof of CRT (5)

More efficient constructive proof.

Reference

“The Residue Number System” by Garner, 1959.

Reference

“The Art of Computer Programming, Vol 2: Seminumerical Algorithms (Section 4.3.2)” by Donald E. Knuth, 3rd edition.



Operations over CRT

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

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

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

Operations over 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

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

Operations over 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.

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

Operations over 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.

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



The ϕ function

Theorem (The ϕ function)

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

The ϕ function

Theorem (The ϕ function)

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

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

The ϕ function

Theorem (The ϕ function)

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

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

“We shall not prove this formula here.” — CLRS (Section 31.3)

The ϕ function

Theorem (The ϕ function)

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

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

Let us prove this formula now.

The ϕ function

Theorem (The ϕ function)

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

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

Let us prove this formula now.

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

The ϕ function

Theorem (The ϕ function)

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

The ϕ function

Theorem (The ϕ function)

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

Proof.

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

$$U_m = \{b \bmod m, (b, m) = 1\} \quad U_n = \{c \bmod n, (c, n) = 1\}$$

The ϕ function

Theorem (The ϕ function)

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

Proof.

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

$$U_m = \{b \bmod m, (b, m) = 1\} \quad U_n = \{c \bmod n, (c, n) = 1\}$$

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

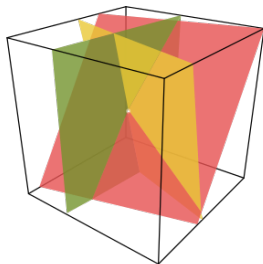
$$f(a \bmod mn) = (a \bmod m, a \bmod n).$$



Secret sharing using the CRT

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

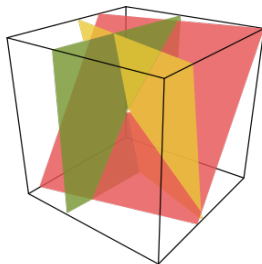
$(3, 3)$ -secret sharing:



Secret sharing using the CRT

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

$(3, 3)$ -secret sharing:



Reference

“How to Share a Secret” by Maurice 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 simultaneous 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 simultaneous congruences

(TC 31.5–2)

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

$$x \equiv 10 \pmod{504}$$

Solving simultaneous congruences

CRT with large modulus

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

Solving simultaneous congruences

CRT with large modulus

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

$$\begin{cases} 19x \equiv 556 \pmod{3} \\ 19x \equiv 556 \pmod{5} \\ 19x \equiv 556 \pmod{7} \\ 19x \equiv 556 \pmod{11} \end{cases}$$

Solving simultaneous congruences

CRT with large modulus

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

$$\left\{ \begin{array}{l} 19x \equiv 556 \pmod{3} \\ 19x \equiv 556 \pmod{5} \\ 19x \equiv 556 \pmod{7} \\ 19x \equiv 556 \pmod{11} \end{array} \right. \quad \left\{ \begin{array}{l} x \equiv 1 \pmod{3} \\ x \equiv 4 \pmod{5} \\ x \equiv 2 \pmod{7} \\ x \equiv 9 \pmod{11} \end{array} \right.$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \end{cases}$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \\ x \equiv 1 \pmod{5} \end{cases} \quad \begin{cases} x \equiv 3 \pmod{2^2} \\ x \equiv 1 \pmod{5} \end{cases}$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \end{cases} \quad \begin{cases} x \equiv 3 \pmod{2^2} \\ x \equiv 1 \pmod{5} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \\ x \equiv 1 \pmod{5} \end{cases}$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \end{cases} \quad \begin{cases} x \equiv 3 \pmod{2^2} \\ x \equiv 1 \pmod{5} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \\ x \equiv 1 \pmod{5} \end{cases}$$

$$\begin{cases} x \equiv 3 \pmod{2^3} \\ x \equiv 3 \pmod{2^2} \end{cases}$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \end{cases} \quad \begin{cases} x \equiv 3 \pmod{2^2} \\ x \equiv 1 \pmod{5} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \\ x \equiv 1 \pmod{5} \end{cases}$$

$$\begin{cases} x \equiv 3 \pmod{2^3} \\ x \equiv 3 \pmod{2^2} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \end{cases}$$

Solving simultaneous congruences

CRT with non-pairwisely co-prime moduli

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

$$\begin{cases} x \equiv 3 \pmod{2^3} \end{cases} \quad \begin{cases} x \equiv 3 \pmod{2^2} \\ x \equiv 1 \pmod{5} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \\ x \equiv 1 \pmod{5} \end{cases}$$

$$\begin{cases} x \equiv 3 \pmod{2^3} \\ x \equiv 3 \pmod{2^2} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{3} \end{cases} \quad \begin{cases} x \equiv 1 \pmod{5} \\ x \equiv 1 \pmod{5} \end{cases}$$

Solving simultaneous congruences

Theorem (CRT with non-pairwisely coprime moduli)

$$a_i \equiv a_j \pmod{(n_i, n_j)}$$

$$0 \leq a < \text{lcm}(n_1, n_2, \dots, n_k)$$

Simultaneous incongruences

$$\exists a, \forall 1 \leq i \leq k : a \not\equiv a_i \pmod{n_i}$$

Simultaneous incongruences

$$\exists a, \forall 1 \leq i \leq k : a \not\equiv a_i \pmod{n_i}$$

NP-complete

