Zastosowania arytmetyki modularnej

Przykład: obliczanie $z = x + y = 123684 + 413456$ na komputerze przyjmującym słowa o długości 100

Obliczamy kongruencje:

$$\begin{aligned} x &\equiv 33 \pmod{99}, & y &\equiv 32 \pmod{99}, \\ x &\equiv 8 \pmod{98}, & y &\equiv 92 \pmod{98}, \\ x &\equiv 9 \pmod{97}, & y &\equiv 42 \pmod{97}, \\ x &\equiv 89 \pmod{95}, & y &\equiv 16 \pmod{95}, \end{aligned}$$

co daje

$$\begin{aligned} z = x + y &\equiv 65 \pmod{99}, \\ z = x + y &\equiv 2 \pmod{98}, \\ z = x + y &\equiv 51 \pmod{97}, \\ z = x + y &\equiv 10 \pmod{95}. \end{aligned}$$

z chińskiego twierdzenia o resztach

$$x + y = z = \sum_{i=1}^{4} z_i M_i {M_i}^{-1} \pmod{M} \quad \text{gdzie}$$

$$M = m_1 m_2 m_3 m_4, \quad M_i = \frac{M}{m_i}, \quad M_i {M_i}^{-1} \equiv 1 \pmod{m_i}$$

Obliczamy:

$M = 99 \cdot 98 \cdot 97 \cdot 95 = 89403930$; $M_1 = M/99 = 903070$, $M_2 = M/98 = 912285$, $M_3 = M/97 = 921690$, $M_4 = M/95 = 941094$,

i rozwiązujemy 4 kongruencje

$$\begin{aligned} 903070 {M_1}^{-1} &\equiv 91 {M_1}^{-1} \equiv 1 \pmod{99}, \\ 912285 {M_2}^{-1} &\equiv \ \ 3 {M_2}^{-1} \equiv 1 \pmod{98}, \\ 921690 {M_3}^{-1} &\equiv 93 {M_3}^{-1} \equiv 1 \pmod{97}, \\ 941094 {M_4}^{-1} &\equiv 24 {M_4}^{-1} \equiv 1 \pmod{95} \end{aligned}$$

${M_1}^{-1} \equiv 37 \pmod{99}$, ${M_2}^{-1} \equiv 33 \pmod{98}$, ${M_3}^{-1} \equiv 24 \pmod{97}$, $M_4^{-1} \equiv 4 \pmod{95}$,

$z \equiv \sum_{i=1}^{4} z_i M_i {M_i}^{-1} \pmod{M} = \ldots = 537140 \pmod{89403930}$.

Obliczenia w systemach resztowych, c.d. [Song Y. Yan]

Przykład: obliczanie $z = x \cdot y = 123684 \cdot 413456$ na komputerze przyjmującym słowa o długości 100

Obliczamy kongruencje:

$$\begin{aligned}
x &\equiv 33 \pmod{99}, & y &\equiv 32 \pmod{99},\\
x &\equiv 8 \pmod{98}, & y &\equiv 92 \pmod{98},\\
x &\equiv 9 \pmod{97}, & y &\equiv 42 \pmod{97},\\
x &\equiv 89 \pmod{95}, & y &\equiv 16 \pmod{95},\\
x &\equiv 63 \pmod{89}, & y &\equiv 51 \pmod{89},\\
x &\equiv 14 \pmod{83}, & y &\equiv 33 \pmod{83}.
\end{aligned}$$

co daje

$$\begin{aligned}
z = x \cdot y &\equiv 66 \pmod{99}, & z = x \cdot y &\equiv 10 \pmod{95},\\
z = x \cdot y &\equiv 50 \pmod{98}, & z = x \cdot y &\equiv 9 \pmod{89},\\
z = x \cdot y &\equiv 51 \pmod{97}, & z = x \cdot y &\equiv 47 \pmod{83}.
\end{aligned}$$

chińskie twierdzenie o resztach ($M = m_1 \ldots m_6 = 803\,651\,926\,770$)

$z = x \cdot y = 51\,137\,891\,904$

Wykrywanie błędów w łańcuchach liczb (bitów)
Song Y. Yan

Bit (sprawdzania) parzystości. . .

to bit x_{n+1} dodany do wysyłanego łańcucha $x_1x_2\ldots x_n$.

w języku kongruencji $x_{n+1} \equiv x_1 + x_2 + \ldots + x_n \pmod 2$
– $x_{n+1} = 0$ dla parzystej liczby jedynek w łańcuchu $x_1x_2\ldots x_n$;
– $x_{n+1} = 1$ dla nieparzystej liczby jedynek.
$\boxed{x_1x_2\ldots x_{n+1} \Longrightarrow y_1y_2\ldots y_{n+1}}$; sprawdzenie to weryfikacja:
$y_1 + y_2 + \ldots + y_{n+1} \equiv 0 \pmod 2$
Ta idea (także w implementacji „macierzowej”) znajduje naturalne rozszerzenie w weryfikacji przeróżnych „numerów”:
social security number,
numery kont bankowych IBAN (*International Bank Account Number*),
UPC (*universal product number*),
numery banknotów
a także numery ISBN (*international standard book number.*)

Wykrywanie błędów w numerach ISBN [Song Y. Yan]

ISBN to 10-cyfrowy numer: $x_1x_2\ldots x_9$; gdzie $x_i \in [0,9]$ oraz *„cyfra" sprawdzająca* $x_{10} \in [0,\ldots,9,X]$, gdzie X oznacza liczbę 10. Używamy tu kongruencji modulo 11: kongruencja weryfikacyjna to

$$\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}, \quad \rightarrow \quad x_{10} \equiv \sum_{i=1}^{9} ix_i \pmod{11}.$$

Przykład: ISBN ma postać 0-387-97329-?

$x_{10} = \sum_{i=1}^{9} ix_i \equiv 1{\cdot}0+2{\cdot}3+3{\cdot}8+4{\cdot}7+5{\cdot}9+6{\cdot}7+7{\cdot}3+8{\cdot}2+9{\cdot}9 \equiv 10 = X.$

Kompletny ISBN ma postać 0-387-97329-X.

Wykrywanie błędów w numerach ISBN, c.d.

Przykład: ISBN ma postać 9-810-x3422-9

aby znaleźć x rozpisujemy $\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$:
$1 \cdot 9 + 2 \cdot 8 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot x + 6 \cdot 3 + 7 \cdot 4 + 8 \cdot 2 + 9 \cdot 2 + 10 \cdot 8 \equiv 1 + 5x \pmod{11}$.
Do rozwiązania mamy kongruencję

$$1 + 5x \equiv 0 \pmod{11}, \qquad x = 2.$$

Zauważmy, że kod ISBN nie zawiera w sobie mechanizmu autokorekty – jedynie można wykryć, że jest błędny (chyba, że dokładnie wiemy która cyfra jest błędna).

Wykrywanie błędów w numerach IBAN [Wikipedia, PL]

schemat – PLkk BBBB BBBk MMMM MMMM MMMM MMMM

- Pierwsze 2 cyfry to cyfry kontrolne.
- Następnych 8 cyfr to identyfikator banku (pierwsze 3 lub 4) i oddziału (następne 4 lub 3).
- Ostatnie 16 cyfr to numer rachunku.

Sprawdzanie i wyliczanie cyfr kontrolnych

- Weź pełen numer konta (razem z kodem kraju), bez spacji. Sprawdź czy zgadza się długość numeru dla danego kraju.
- Przenieś 4 pierwsze znaki numeru na jego koniec.
- Zamień litery w numerze konta na cyfry, zamieniając 'A' na 10, 'B' na 11 itd., aż do 'Z' na 35.
- Potraktuj otrzymany ciąg znaków jak liczbę i wylicz jej resztę modulo 97. Jeśli reszta jest równa 1, to numer konta ma prawidłowe cyfry kontrolne.

Liniowy generator kongruentny (LCG) . . .

. . . został wprowadzony w latach 1950-tych przez D.H. Lehmera – ojca obliczeniowej teorii liczb.

LCG Lehmera:

(1) n — moduł $(n > 0)$; (2) x_0 — ziarno $(0 \leqslant x_0 \leqslant n)$;
(3) a — mnożnik $(0 \leqslant a \leqslant n)$; (4) b — przesunięcie $(0 \leqslant b \leqslant n)$.

$$\boxed{x_k \equiv a x_{k-1} + b \pmod{n}, \quad 1 \leqslant k \leqslant K,}$$

– K – długość okresu.

Jasnym jest, że długość okresu K nie może przekraczać wartości modułu n; warunek konieczny aby K było równe n (Knuth) to:
$(b, n) = 1$; $a \equiv 1 \pmod{p}$ dla wszystkich $p \mid n$,
a także $a \equiv 1 \pmod{4}$ jeżeli $4 \mid n$.
Dla $a = 1$ mówimy o *czystym liniowym generatorem kongruentnym*;
dla $a > 1$ – *multyplikatywnym liniowym generatorem kongruentnym*.

LCG przykład:

(1) $n = 1399$ — moduł; (2) $x_0 = 5$ — ziarno; (3) $a = 11$ — mnożnik; (4) $b = 73$ — przesunięcie; (5) k – liczba liczb losowych.

$$
\begin{aligned}
x_1 &\equiv ax_0 + b \pmod{n} \rightarrow x_1 = 128, & x_6 &\equiv ax_5 + b \pmod{n} \rightarrow x_6 = 768,\\
x_2 &\equiv ax_1 + b \pmod{n} \rightarrow x_2 = 82, & x_7 &\equiv ax_6 + b \pmod{n} \rightarrow x_7 = 127,\\
x_3 &\equiv ax_2 + b \pmod{n} \rightarrow x_3 = 975, & x_8 &\equiv ax_7 + b \pmod{n} \rightarrow x_8 = 71,\\
x_4 &\equiv ax_3 + b \pmod{n} \rightarrow x_4 = 1005, & x_9 &\equiv ax_8 + b \pmod{n} \rightarrow x_9 = 854,\\
x_5 &\equiv ax_4 + b \pmod{n} \rightarrow x_5 = 1335, & x_{10} &\equiv ax_9 + b \pmod{n} \rightarrow x_{10} = 1073\\
&\ldots\ldots & &\ldots\ldots\\
x_{232} &\equiv ax_{231} + b \pmod{n} \rightarrow x_{232} = 121,\\
x_{233} &\equiv ax_{232} + b \pmod{n} \rightarrow x_{233} = 5.
\end{aligned}
$$

Długość ciągu $K = 233$.

Często stosowane wartości $n = 2^m$, $a = 2^l + 1$, $(l < m)$, $b = 1$.

Generator kongruentny potęgowy . . .

. . . ma postać ogólną

$$x_k \equiv (x_{k-1})^d \pmod{n}, \quad k = 1, 2, \ldots$$

Generator potęgowy RSA

$n = pq$ oraz $(d, \phi(n)) = 1$. Na przykład:
$n = 299 = 13 \cdot 23;\ \phi(299) = 264;\ d = 17\ (17, 264) = 1;\ x_0 = 6$

$K = 10$

Generator kwadratowy $d = 2$

$n = pq$ gdzie $p \equiv q \equiv 3 \pmod{4}$.

Dyskretny generator wykładniczy

$$x_k \equiv g^{x_{k-1}} \pmod{n},$$

Jeżeli $n = p > 2$, a g jest pierwiastkiem pierwotnym modulo p to algorytm „wykładniczy” zastosowany wstecz $x_k \to x_{k-1}$ jest po prostu zagadnieniem logarytmów dyskretnych.