# *Projektowanie generatorów kongruencyjnych, wytwarzających ciągi okresowe liczb naturalnych*

*Andrzej Paszkiewicz*

*Omówiono podstawowe zasady projektowania generatorów kongruencyjnych liczb pseudolosowych, ze szczególnym uwzględnieniem generatorów afinicznych i inwersyjnych.*

***generatory liczb pseudolosowych, pierwiastki pierwotne***

## Generatory liczb pseudolosowych

Istnieje kilka powodów, dla których warto zająć się generowaniem w sposób deterministyczny ciągów liczb mających własności zbliżone do losowych. Trzy spośród nich są najistotniejsze.

1. Ciągi te wykorzystuje się w procesach symulacji cyfrowej. Determinizm występowania kolejnych znaków (liczb) generowanego ciągu jest w tym przypadku korzystny, gdyż można eksperyment symulacyjny powtórzyć bez potrzeby pamiętania całej wygenerowanej sekwencji pseudolosowej. Do odtworzenia ciągu wystarczy zapamiętać tylko niewielką ilość informacji – warunki startowe wykorzystywanego generatora, czyli tzw. ziarno.
2. Ciągi pseudolosowe są używane do wykonywania obliczeń z wykorzystaniem metod Monte-Carlo. Obliczenia numeryczne metodami Monte-Carlo są niekiedy jedyną metodą rozwiązywania skomplikowanych problemów naukowo-technicznych. Przykładem mogą być równania różniczkowe cząstkowe czy nieliniowe równania różniczkowe zwyczajne, pojawiające się w modelach prognozowania pogody lub modelach wyznaczania trajektorii pojazdów kosmicznych. W tym przypadku możliwość powtórzenia obliczeń przez kogoś innego, na niezależnej platformie sprzętowej, sprawia, że ciągi pseudolosowe będą lepsze od losowych.
3. Na jakość ciągów pseudolosowych można mieć wpływ. Cechy tej nie mają ciągi losowe, których jakość można jedynie badać. Właśnie z tego powodu generatory ciągów pseudolosowych są chętnie wykorzystywane w kryptografii. Ona też jest ich największym odbiorcą.

Wytwarzane w sposób algorytmiczny ciągi pseudolosowe wykorzystywane w praktyce są ciągami okresowymi. Długość okresu ciągu pseudolosowego jest istotnym parametrem decydującym o jego jakości. W pewnym uproszczeniu można przyjąć, że im większa długość okresu generowanego ciągu, tym jest on lepszy. Nie jest to jednak warunek wystarczający. Nie wystarczy stwierdzić, że okres generowanego ciągu jest duży, aby twierdzić, że jest on wystarczająco „dobry" dla wszystkich zastosowań.

**Przykład 1**

Weźmy nieskończony ciąg kolejnych liczb naturalnych, ograniczając się jedynie do 32 najmniej znaczących bitów każdej z liczb. Jest to ciąg okresowy o długości okresu równej $2^{32}$. Okres tego ciągu jest duży, jednak jakość tak generowanego ciągu jest dla większości zastosowań niewystarczająca.

Miary jakości ciągów pseudolosowych można podzielić na algebraiczne i statystyczne. Istnieją również pewne miary jakości ciągów pseudolosowych wyprowadzone z teorii informacji. W tym artykule nie będzie jednak rozwijany problem oceny jakości ciągów pseudolosowych, natomiast zostaną szczegółowo omówione najprostsze generatory kongruencyjne ciągów pseudolosowych, a zwłaszcza:

- generatory liniowe i afiniczne;
- generator kwadratowy i generatory wielomianowe, których własności dają się wyprowadzić z teorii tzw. wielomianów permutacyjnych;
- generatory inwersyjne.

# Liniowe i afiniczne generatory kongruencyjne

Przy generowaniu ciągów pseudolosowych często dąży się (np. z powodów czasowych), aby proces generowania tych ciągów cechował się jak największą prostotą. Do najprostszych generatorów ciągów pseudolosowych można zaliczyć generatory liniowe i afiniczne.

## *Liniowe generatory kongruencyjne*

Liniowy generator kongruencyjny jest opisany za pomocą zależności

$$X_{n+1} = g \cdot X_n \,(\mathrm{mod}\ M) \tag{1}$$

dla $n = 0, 1, 2, \ldots,$ gdzie $X_0, g \in \{0, 1, \ldots, M-1\}$.

Generator (1) został zaproponowany w 1951 r. przez D.H. Lehmera. Jest to generator okresowy, niezwykle efektywny, gdy $M$ jest potęgą liczby 2. W takim przypadku redukcja modularna jest automatyczna. Parametry tego generatora, jeśli $M = 2^m$, określa twierdzenie 1.

**Twierdzenie 1**

Jeśli $M = 2^m$, $m \geqslant 3$, wówczas maksymalny okres generatora liniowego (1) wynosi $M = 2^{m-2}$. Okres ten jest osiągalny wtedy i tylko wtedy, gdy $X_0$ jest liczbą nieparzystą, reszta zaś z dzielenia liczby $g$ przez 8 wynosi 3 lub 5, tzn.:

$$g \equiv 3 \ (\mathrm{mod}\ 8) \quad \text{lub} \quad g \equiv 5 \ (\mathrm{mod}\ 8)\,.$$

Można stwierdzić, że, gdy $M$ jest potęgą dwójki, operacja „obcięcia modularnego", tj. brania reszty modulo $M$, „nic nie kosztuje". Kolejny wyraz ciągu o numerze $n+1$ można otrzymać mnożąc wyraz bieżący o numerze $n$ przez stały czynnik $g$ i biorąc liczbę mieszczącą się na $m$ najmłodszych bitach iloczynu. Generator ten jest łatwy w implementacji i szybki.

**Przykład 2**

Weźmy $m = 5$. Zgodnie z twierdzeniem 1, jako $g$ występujące w definicji generatora liniowego można przyjąć następujące liczby: 3, 11, 19, 27, 5, 13, 21, 29. Dla każdego $g$ spośród wypisanych liczb otrzymujemy następujące ciągi:

$g = 3$: 1, 3, 9, 27, 17, 19, 25, 11
$g = 11$: 1, 11, 25, 19, 17, 27, 9, 3
$g = 19$: 1, 19, 9, 11, 17, 3, 25, 27
$g = 27$: 1, 27, 25, 3, 17, 11, 9, 19
$g = 5$: 1, 5, 25, 29, 17, 21, 9, 13
$g = 13$: 1, 13, 9, 21, 17, 29, 25, 5
$g = 21$: 1, 21, 25, 13, 17, 5, 9, 29
$g = 29$: 1, 29, 9, 5, 17, 13, 25, 21

Warto zwrócić uwagę, że kolejnymi wyrazami ciągu (1), w przypadku gdy $M$ jest potęgą liczby 2, są wybrane liczby nieparzyste. Najmłodszy bit każdego z wyrazów ciągu jest zatem nieistotny (na stałe równy 1). Mankamentu tego nie mają generatory liniowe, w których liczba modularna $M$ jest liczbą pierwszą.

**Twierdzenie 2**

Jeśli $M = p$, gdzie $p$ jest liczbą pierwszą, wówczas generator liniowy (1) ma okres maksymalny równy $p$. Okres ten osiąga się, gdy $g$ jest pierwiastkiem pierwotnym liczby $p$.

Szukanie pierwiastka pierwotnego liczby pierwszej $p$ wiąże się z pewną uciążliwością (zwłaszcza dla dużych liczb $p$), jednak stosunek długości okresu do liczby modularnej jest w tym przypadku zdecydowanie korzystniejszy, niż gdy liczba modularna $M$ ma postać $2^m$.

Należy przedstawić teraz algorytm znajdowania najmniejszego pierwiastka pierwotnego liczby pierwszej $p$. Niech liczba $p-1$ ma rozkład kanoniczny:

$$p - 1 = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}. \tag{2}$$

---

**Algorytm znajdowania najmniejszego pierwiastka pierwotnego**

*Dane*: liczba pierwsza $p$, znany rozkład kanoniczny $p-1$.

START

Krok 1: $g := 1$.

Krok 2: $g := g + 1$.
Dopóki $g$ jest potęgą liczby naturalnej, należy wykonać $g := g + 1$.

Krok 3: Jeśli istnieje $1 \leqslant i \leqslant k$, dla którego $g^{\frac{p-1}{p_i}} \equiv 1 \pmod{p}$, wówczas należy przejść do kroku 2.

Krok 4: Znaleziona wartość $g$ jest najmniejszym pierwiastkiem pierwotnym liczby pierwszej $p$. STOP

---

Procedura znajdowania najmniejszego pierwiastka pierwotnego liczby pierwszej $p$ jest skuteczna. Jak wskazują wyniki obliczeń numerycznych (tablica 1), liczby $g$ są bardzo małe w stosunku do $p$.

***Tabl. 1. Częstość występowania liczb naturalnych mniejszych od 100 jako najmniejsze pierwiastki pierwotne liczb pierwszych***

| $n$ | $\sum_{\substack{p < 25\cdot 10^9 \\ g(p)=n}} 1$ | $\frac{1}{\pi(25\cdot 10^9)} \sum_{\substack{p < 25\cdot 10^9 \\ g(p)=n}} 1$ | $n$ | $\sum_{\substack{p < 25\cdot 10^9 \\ g(p)=n}} 1$ | $\frac{1}{\pi(25\cdot 10^9)} \sum_{\substack{p < 25\cdot 10^9 \\ g(p)=n}} 1$ |
|---|---|---|---|---|---|
| 2 | 408356348 | 0,373957012810 | 52 | 10932 | 0,000010011104 |
| 3 | 247450040 | 0,226605214371 | 53 | 117886 | 0,000107955458 |
| 5 | 151852967 | 0,139061097504 | 54 | 20 | 0,000000018315 |
| 6 | 61024090 | 0,055883510854 | 55 | 18543 | 0,000016980965 |
| 7 | 75027518 | 0,068707310777 | 56 | 349 | 0,000000319601 |
| 10 | 25201874 | 0,023078905384 | 57 | 27252 | 0,000024956332 |
| 11 | 40661009 | 0,037235785700 | 58 | 37410 | 0,000034258637 |
| 12 | 3562957 | 0,003262818769 | 59 | 62653 | 0,000057375204 |
| 13 | 25374790 | 0,023237255195 | 60 | 26 | 0,000000023810 |
| 14 | 9029930 | 0,008269262043 | 61 | 44630 | 0,000040870435 |
| 15 | 4586427 | 0,004200073168 | 62 | 16586 | 0,000015188820 |
| 17 | 12630399 | 0,011566432856 | 63 | 378 | 0,000000346158 |
| 18 | 441977 | 0,000404745511 | 65 | 6311 | 0,000005779371 |
| 19 | 8281573 | 0,007583945531 | 66 | 1270 | 0,000001163017 |
| 20 | 184208 | 0,000168690590 | 67 | 26072 | 0,000023875733 |
| 21 | 1747932 | 0,001600688792 | 68 | 2116 | 0,000001937751 |
| 22 | 2685865 | 0,002459611702 | 69 | 6617 | 0,000006059594 |
| 23 | 4228359 | 0,003872168288 | 70 | 514 | 0,000000470701 |
| 24 | 24820 | 0,000022729200 | 71 | 16689 | 0,000015283143 |
| 26 | 1416534 | 0,001297207270 | 73 | 12110 | 0,000011089871 |
| 28 | 164635 | 0,000150766391 | 74 | 4756 | 0,000004355362 |
| 29 | 2406740 | 0,002203999780 | 75 | 17 | 0,000000015568 |
| 30 | 114954 | 0,000105270445 | 76 | 879 | 0,000000804954 |
| 31 | 1616424 | 0,001480258831 | 77 | 1241 | 0,000001136460 |
| 33 | 378134 | 0,000346280551 | 78 | 415 | 0,000000380041 |
| 34 | 507785 | 0,000465009942 | 79 | 7007 | 0,000006416741 |
| 35 | 165354 | 0,000151424823 | 82 | 2734 | 0,000002503692 |
| 37 | 825083 | 0,000755579228 | 83 | 4390 | 0,000004020193 |
| 38 | 287238 | 0,000263041495 | 84 | 2 | 0,000000001832 |
| 39 | 168011 | 0,000153858002 | 85 | 852 | 0,000000780229 |
| 40 | 3487 | 0,000003193260 | 86 | 1649 | 0,000001510091 |
| 41 | 450745 | 0,000412774907 | 87 | 1045 | 0,000000956971 |
| 42 | 11547 | 0,000010574298 | 88 | 8 | 0,000000007326 |
| 43 | 312095 | 0,000285804578 | 89 | 2332 | 0,000002135556 |
| 44 | 20843 | 0,000019087216 | 91 | 310 | 0,000000283886 |
| 45 | 868 | 0,000000794881 | 92 | 212 | 0,000000194141 |
| 46 | 107946 | 0,000098852789 | 93 | 592 | 0,000000542131 |
| 47 | 186811 | 0,000171074317 | 94 | 784 | 0,000000717957 |
| 48 | 26 | 0,000000023810 | 95 | 335 | 0,000000306780 |
| 50 | 96 | 0,000000087913 | 97 | 1230 | 0,000001126387 |
| 51 | 46354 | 0,000042449208 | 99 | 16 | 0,000000014652 |

Oznaczenia: $\pi(x)$ – liczba liczb pierwszych mniejszych od $x$.

Również wartość średnia najmniejszego pierwiastka pierwotnego liczby pierwszej wydaje się dążyć do stałej.

Do tej pory nie wyjaśniano w tym artykule, czy każda liczba pierwsza ma pierwiastek pierwotny. Odpowiedź na to pytanie jest pozytywna. Świadczy o tym twierdzenie 3.

**Twierdzenie 3**

Istnieje $\varphi(p-1)$ pierwiastków pierwotnych liczby pierwszej $p$. Jeśli $g$ jest dowolnym pierwiastkiem pierwotnym modulo $p$ oraz $(k, p-1)=1$ dla liczby $k$, to również $g^k$ jest pierwiastkiem pierwotnym modulo $k$. Dla liczb naturalnych $x$, $y$ symbol $\varphi(x)$ oznacza funkcję Eulera, $(x, y)$ zaś największy wspólny dzielnik liczb $x$, $y$.

Znajdowanie pierwiastków pierwotnych zilustrowano w przykładzie 3.

**Przykład 3**

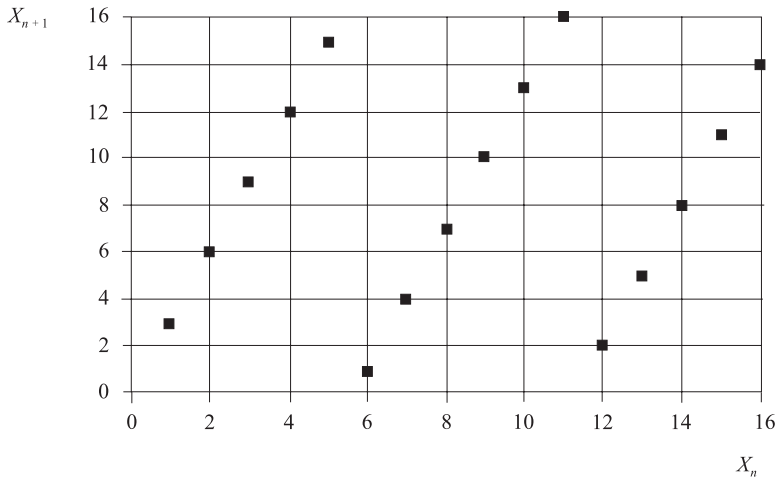
Weźmy liczbę pierwszą $p=71$ i spróbujmy znaleźć jej najmniejszy pierwiastek pierwotny. Łatwo można uzyskać rozkład kanoniczny liczby $p-1=70=2\cdot 5\cdot 7$. Zgodnie z algorytmem, trzeba liczby naturalne podnosić do potęg 10, 14 oraz 35 i sprawdzać, czy otrzymane liczby w wyniku dzielenia przez 71 dają wszystkie reszty różne od 1. Jest $2^{10} \pmod{71} \equiv 30$, $2^{14} \pmod{71} \equiv 54$, $2^{35} \pmod{71} \equiv 1$. Ze względu na ostatnią resztę (równą 1) liczba 2 nie jest pierwiastkiem pierwotnym liczby 71. Należy wziąć liczbę 3 i sprawdzić, czy jej potęgi 10., 14. i 35. dają w wyniku dzielenia przez 71 reszty różne od 1. Jest $3^{10} \pmod{71} \equiv 48$, $3^{14} \pmod{71} \equiv 54$, $3^{35} \pmod{71} \equiv 1$. Ze względu na ostatnią resztę równą 1, liczba 3 nie może być pierwiastkiem pierwotnym liczby 71. Ponieważ liczba 4 jest potęgą liczby naturalnej, więc – zgodnie z algorytmem – pomija się ją jako kandydatkę na najmniejszy pierwiastek pierwotny. Teraz należy zbadać liczbę 5. Jest $5^{10} \pmod{71} \equiv 1$, zatem 5 nie może być pierwiastkiem pierwotnym liczby 71. Należy sprawdzić, czy liczba 6 jest pierwiastkiem pierwotnym liczby 71. Można skorzystać z poprzednio uzyskanych wyników. Jest: $6^{10} \pmod{71} \equiv 2^{10}\cdot 3^{10} \pmod{71} \equiv 30\cdot 48 \pmod{71} \equiv 20$, $6^{14} \pmod{71} \equiv 2^{14}\cdot 3^{14} \pmod{71} \equiv 54\cdot 54 \pmod{71} \equiv 5$, $6^{35} \pmod{71} \equiv 2^{35}\cdot 3^{35} \pmod{71} \equiv 1\cdot 1 \pmod{71} \equiv 1$. Tak więc liczba 6 również nie może być pierwiastkiem pierwotnym liczby 71. Należy sprawdzić, czy liczba 7 jest odpowiednią kandydatką na najmniejszy pierwiastek pierwotny liczby 71. Uzyskuje się $7^{10} \pmod{71} \equiv 45$, $7^{14} \pmod{71} \equiv 54$, $7^{35} \pmod{71} \equiv 70$. Ostatecznie zatem $g=7$ jest najmniejszym pierwiastkiem pierwotnym liczby 71. Zgodnie z twierdzeniem 3, również $7^k \pmod{71}$ dla $k \in \{1, 3, 9, 11, 13, 17, 19, 23, 27, 29, 31, 33, 37, 39, 41, 43, 47, 51, 53, 57, 59, 61, 67, 69\}$ jest pierwiastkiem pierwotnym liczby 71.

Generatory liniowe mają interesującą własność zauważoną przez Marsaglię [12]. Mianowicie, jeśli wykona się wykres generatora liniowego jako funkcję poprzedniego wyrazu $X_{n+1}=f(X_n) \pmod{M}$, wówczas można zaobserwować, że wyraz $X_{n+1}$ generatora liniowego w zależności od $X_n$ będzie przedstawiał funkcję przedziałami liniową.

**Przykład 4**

Dla $p=17$ liczba $g=3$ jest najmniejszym pierwiastkiem pierwotnym. Wykres zależności $X_{n+1}=f(X_n) \pmod{17}$ dla generatora $X_{n+1}=3\cdot X_n \pmod{17}$, z warunkiem początkowym $X_1=1$, zobrazowano na rys. 1.

Jak można się spodziewać, generatory takie są łatwo przewidywalne, co oznacza, że nie mogą być one bezpośrednio wykorzystywane w aplikacjach związanych z ochroną danych. Z powodzeniem można je natomiast stosować, np. do celów symulacji komputerowej. Wystarczy jednak niezbyt skomplikowana modyfikacja kolejnych wyrazów generowanego ciągu pseudolosowego, aby uzyskał

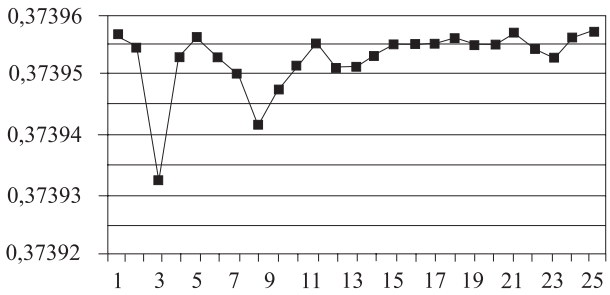



***Rys. 1.*** *Wykres liniowego generatora kongruencyjnego* $x[n] = 3x[n-1] \pmod{17}$

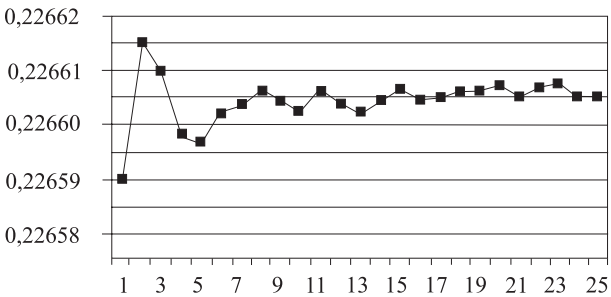
on własności predystynujące go do bardziej wyrafinowanych zastosowań niż symulacja komputerowa. W Politechnice Warszawskiej wykonywano badania statystyczne jakości ciągów z liniowego generatora kongruencyjnego przekształcane za pomocą szyfru DES oraz IDEA, traktowanych jako tzw. funkcja filtrująca. Wyniki tych badań są bardzo obiecujące. Do badań wykorzystywano liczby pierwsze długości 64 bitów w zapisie binarnym. Liczby te tak dobierano, aby miały najmniejszy pierwiastek pierwotny równy 2, 3, 5, 6 lub 7. W takim przypadku generowanie kolejnych wyrazów generatora kongruencyjnego jest łatwe. Mnożenie długiej liczby naturalnej przez liczbę mającą kilka bitów może być łatwo zoptymalizowane. Nie stwierdzono istotnych różnic statystycznych, związanych z faktem wykorzystywania różnych najmniejszych pierwiastków pierwotnych, jak też różnych szyfrów blokowych spośród dwóch wymienionych. Eksperyment był kontynuowany dalej w następujący sposób. Kolejno wygenerowane wyrazy z generatora kongruencyjnego, przekształcone za pomocą szyfru blokowego, poddawano „kompresji" przez odrzucenie najmłodszego bitu, a następnie używano do 63-bitowego wektora funkcji „32 z 63". Jeśli wśród współrzędnych 63-bitowego wektora większość stanowiły „jedynki", to na wyjściu funkcji kompresji przyjmowano wartość 1, w przeciwnym zaś przypadku 0. Na kilkaset przebadanych próbek tylko kilka z nich nie przeszło testu serii. Po zsumowaniu modulo 2 wygenerowanego ciągu z ciągiem z rejestru liniowego ze sprzężeniem zwrotnym własności statystyczne badanych próbek uległy poprawie. Użycie jednego bitu spośród 64 wydaje się mało oszczędne, dlatego też wykorzystywano wszystkie bity, po przekształceniu za pomocą funkcji filtrującej do szyfrowania strumieniowego 64 kanałów. Tak uzyskane ciągi bitów wykazują bardzo małą korelację i mogą być z pewnym przybliżeniem uznane za niezależne źródła bitów. Podobnie jak w poprzednim przypadku, niewielki procent próbek zbieranych z poszczególnych kanałów nie przechodzi testu spektralnego, testu Maurera lub testu serii. Po zsumowaniu modulo 2 wygenerowanego ciągu z ciągiem z rejestru liniowego ze sprzężeniem zwrotnym, o sprzężeniu opisanym przez wielomian pierwotny, własności statystyczne badanych próbek uległy poprawie. Liniowy generator kongruencyjny oraz funkcje filtrujące zaimplementowano programowo oraz sprzętowo w układach firmy ALTERA, doprowadzając projekt do poziomu poprawnej symulacji.

W artykule [11] omówiono badania częstości występowania poszczególnych liczb naturalnych jako najmniejsze pierwiastki pierwotne. Przebadano wszystkie liczby pierwsze mniejsze od $25 \cdot 10^9$.

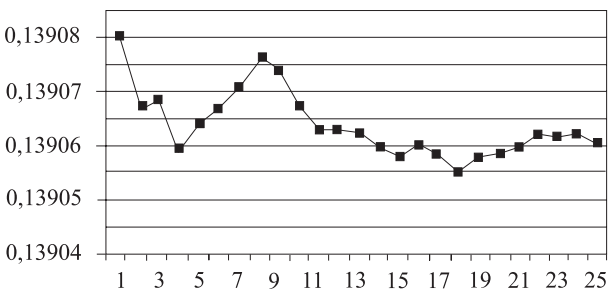
W efekcie stwierdzono, że spośród liczb naturalnych mniejszych od 100 i nie będących potęgami liczb naturalnych jedynie liczby 72, 80, 90, 96 i 98 nie pojawiły się jako najmniejsze pierwiastki pierwotne liczb pierwszych. Zgodnie z twierdzeniem Hooley'a z 1967 r. [6], udowodnionym przy założeniu Wielkiej Hipotezy Riemanna [9] dla funkcji dzeta Dedekinda [9], każda liczba naturalna, nie będąca kwadratem innej liczby naturalnej, pojawia się jako pierwiastek pierwotny liczby pierwszej z niezerową częstością.




***Rys. 2.*** *Częstość występowania liczb pierwszych, mających najmniejszy pierwiastek pierwiastek pierwotny równy 2*



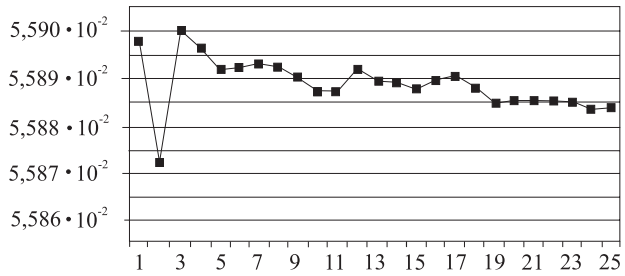

***Rys. 3.*** *Częstość występowania liczb pierwszych, mających najmniejszy pierwiastek pierwiastek pierwotny równy 3*



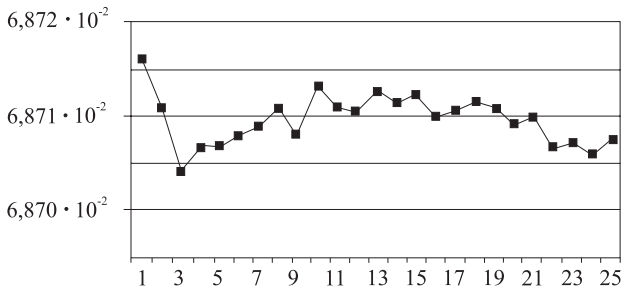

***Rys. 4.*** *Częstość występowania liczb pierwszych, mających najmniejszy pierwiastek pierwiastek pierwotny równy 5*

W pracy [5] Elliott oraz Murata podali wzory, umożliwiające policzenie jak często zadane liczby naturalne występują jako najmniejsze pierwiastki pierwotne liczb pierwszych. Zgodnie z tymi wzorami, częstości $D(n)$ pojawiania się liczb $n$ jako najmniejsze pierwiastki pierwotne dla $n = 2, 3, 5, 6$ oraz 7 są

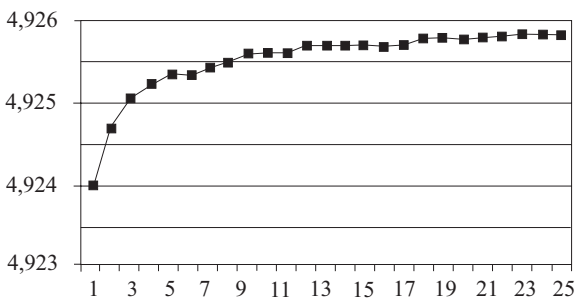
odpowiednio równe: $D(2) = 0,373955$, $D(3) = 0,226606$, $D(5) = 0,139065$, $D(6) = 0,055883$, $D(7) = 0,068707$. Przeprowadzone badania numeryczne w pełni potwierdzają wyniki Elliotta i Muraty [5], uzyskane przy założeniu Wielkiej Hipotezy Riemanna. Zbieżność przewidywań teoretycznych z praktyką pokazano na rys. $2 \div 6$. Wyniki badań zostały stablicowane z krokiem równym $10^9$ (rys. 7).




***Rys. 5.*** *Częstość występowania liczb pierwszych, mających najmniejszy pierwiastek pierwiastek pierwotny równy 6*




***Rys. 6.*** *Częstość występowania liczb pierwszych, mających najmniejszy pierwiastek pierwiastek pierwotny równy 7*




***Rys. 7.*** *Zachowanie się wartości średniej najmniejszego pierwiastka pierwotnego liczby pierwszej, stablicowane w przedziale* $[2..25 \cdot 10^9]$ *z krokiem równym* $10^9$

## *Afiniczne generatory kongruencyjne*

Generatory afiniczne są opisywane za pomocą zależności:

$$X_{n+1} = a \cdot X_n + b \,(\mathrm{mod}\, M), \quad b \neq 0, \quad n = 1,\ 2,\ \ldots\ , \tag{3}$$

gdzie $a,\ b \in \{1, 2, \ldots, M-1\}$.

W przypadku generatora afinicznego jest istotne pytanie, dotyczące długości jego okresu. Odpowiedź na nie stanowi twierdzenie 4.

**Twierdzenie 4**

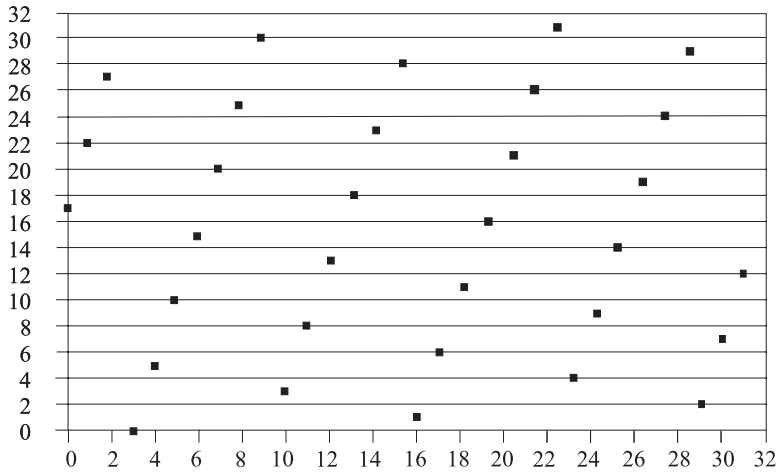
Generator afiniczny (3) ma okres równy $M$ wtedy i tylko wtedy, gdy są spełnione następujące warunki:

1) największy wspólny dzielnik $(b,\ M) = 1$;
2) $a \equiv 1\,(\mathrm{mod\ p})$ dla każdego czynnika pierwszego liczby $M$;
3) $a \equiv 1\,(\mathrm{mod}\ 4)$, jeśli $4 \mid M$.

W szczególnym przypadku, gdy $M = 2^m$, można uzyskać okres pełny równy $2^m$, co nie było możliwe w przypadku liniowego generatora kongruencyjnego o takiej samej długości modułu. Generator afiniczny ma podobne własności jak generator liniowy. Wykres wartości $X_{n+1}$ generatora afinicznego – traktowanej jako funkcja poprzedniego wyrazu $X_n$ – składa się z punktów $(X_n, X_{n+1})$, leżących na kawałkach prostych równoległych.

**Przykład 5**

Przyjmijmy $M = 32$, $a = 5$, $b = 17$. Łatwo sprawdzić, że parametry $M$, $a$, $b$ spełniają warunki twierdzenia 4, zatem generator afiniczny $X_{n+1} = 5 \cdot X_n + 17\,(\mathrm{mod}\ 32)$, z warunkiem początkowym $X_0 = 1$, jest generatorem, dającym pełny okres równy 32. W przypadku tego generatora zbiór par $(X_n, X_{n+1})$ jest następujący {(0, 17), (1, 22), (2, 27), (3, 0), (4, 5), (5, 10), (6, 15), (7, 20), (8, 25),




***Rys. 8.*** *Wykres wartości generatora afinicznego* $X_{n+1} = 5X_n + 17$ *w funkcji* $X_n$

(9, 30), (10, 3), (11, 8), (13, 18), (14, 23), (15, 28), (16, 1), (17, 6), (18, 11), (19, 16), (20, 21), (21, 26), (22, 31), (23, 4), (24, 9), (25, 14), (26, 19), (27, 24), (28, 29), (29, 2), (30, 7), (31, 12)}. Wykres zależności $X_{n+1}$ w funkcji $X_n$ przedstawiono na rys. 8.

Podobnie jak w przypadku generatora liniowego, generator afiniczny nie może być wykorzystywany w bezpośredni sposób we wszelkich aplikacjach, związanych np. z ochroną danych, ze względu na dużą przewidywalność tego generatora. Warto też zwrócić uwagę, że jeśli liczba modularna $M$ jest liczbą bezkwadratową, a więc jeśli liczby pierwsze w jej rozkładzie kanonicznym występują z wykładnikiem równym 1, wówczas osiągnięcie maksymalnego okresu generatora afinicznego jest możliwe wyłącznie wtedy, gdy występująca w jego definicji liczba $a$ jest równa 1.

## *Uogólniony generator liniowy*

Uogólniony generator liniowy można opisać za pomocą następującego równania rekurencyjnego:

$$X_n = a_1 X_{n-1} + \ldots + a_k X_{n-k} \pmod{M}, \quad n = k,\ k+1, \ldots \tag{4}$$

przy ustalonych współczynnikach $a_1, a_2, \ldots, a_k \in \{0, 1, \ldots, M-1\}$ oraz zadanych warunkach początkowych $X_0, X_1, \ldots, X_{k-1} \in \{0, 1, \ldots, M-1\}$, przy czym nie wszystkie $X_i$, $i = 0, 1, \ldots, k$ są jednocześnie równe zero. Najczęściej przyjmujemy $M = p$, gdzie $p$ jest liczbą pierwszą. W takim przypadku można skorzystać z twierdzenia, umożliwiającego takie dobranie współczynników $a_1, a_2, \ldots, a_k \in \{0, 1, \ldots, M-1\}$, aby okres uogólnionego generatora liniowego zadanego formułą (4) był maksymalny.

**Twierdzenie 5**

Uogólniony generator liniowy zadany formułą (4) dla $M = p$, gdzie $p$ jest liczbą pierwszą, ma okres maksymalny równy $p^k - 1$. Okres ten jest osiągany, gdy wielomian

$$f(X) = X^k - a_1 X^{k-1} - \ldots - a_{k-1} X - a_k \tag{5}$$

jest wielomianem pierwotnym modulo $p$.

Dowód tego twierdzenia można znaleźć np. w [10]. Podobnie jak w przypadku generatora liniowego oraz afinicznego, kolejne wyrazy $X_{n-1}$, $i = 0, 1, \ldots, k$ układają się w hiperpłaszczyznach. Generator ten jest, tak jak i poprzednie, łatwo przewidywalny. Znajomość $k$ kolejnych wyrazów umożliwia jednoznaczne wyznaczenie współczynników $a_i$ występujących w równości (4). Generator ten najczęściej jest stosowany dla $p = 2$. Automat generujący sekwencję binarną zgodnie z wzorem (4) nosi wówczas nazwę rejestru liniowego ze sprzężeniem zwrotnym. Istotnym z punktu widzenia zastosowań jest problem znajdowania wielomianów pierwotnych zadanego stopnia $n$ o współczynnikach z zadanego ciała skończonego.

## *Kwadratowy generator kongruencyjny*

Z uwagi na dużą przewidywalność generatorów liniowych i afinicznych prowadzono poszukiwania generatorów bez tego mankamentu. Jedną z propozycji jest generator kwadratowy. Jest on opisany za pomocą następującej zależności:

$$X_{n+1} = aX_n^2 + bX_n + c \pmod{M}, \tag{6}$$

gdzie $a, b, c, X_0 \in \{0, 1, \ldots, M-1\}$, $n = 1, 2, \ldots$ .

Generator ten, podobnie jak generator afiniczny, może wytwarzać ciąg o okresie pełnym równym $M$. Warunki konieczne i wystarczające, aby tak było, zawiera twierdzenie 6.

**Twierdzenie 6**

Kwadratowy generator kongruencyjny ma okres równy $M$ wtedy i tylko wtedy, gdy współczynniki $a, b, c$, występujące we wzorze (6) spełniają następujące warunki:

1) największy wspólny dzielnik $(c, M) = 1$;
2) dla każdej nieparzystej liczby pierwszej $p$, będącej dzielnikiem $M$, jest również $p \mid a$ oraz $p \mid b-1$;
3a) jeśli $4 \mid M$, to $a$ jest liczbą parzystą oraz $a \equiv b-1 \pmod 4$;
3b) jeśli $2 \,||\, M$, to $a \,||\, b-1 \pmod 2$;
4) jeśli $9 \mid M$, to $a \neq 3c \pmod 9$.

Dowód twierdzenia można znaleźć w [10].

Generator ten oraz inne opisywane przez zależność postaci

$$X_{n+1} = \sum_{i=1}^{k} a_i X_n^i \pmod M \tag{7}$$

dla $n = 0, 1, 2, \ldots$, mogą być badane z wykorzystaniem torii wielomianów permutacyjnych [10].

## *Wielomiany permutacyjne i ich własności*

Niech $M$ będzie liczbą naturalną. Wielomian

$$g(X) = \sum_{k=0}^{n} a_k X^k, \quad a_1, a_2, \ldots, a_n \in \{0, 1, \ldots, M-1\} \tag{8}$$

jest wielomianem permutacyjnym modulo $M$ wtedy i tylko wtedy, gdy liczby

$$y = g(X) \pmod M, \quad X \in \{0, 1, \ldots, M-1\} \tag{9}$$

stanowią permutację zbioru $\{0, 1, \ldots, M-1\}$. Przy ustalonej wartości $M$ nie każdy wielomian o współczynnikach ze zbioru $A$ jest wielomianem permutacyjnym tego zbioru. W szczególnym przypadku, jeśli narzuci się wymagania na strukturę zbioru $A$, można podać warunki konieczne i wystarczające istnienia wielomianu permutacyjnego zbioru $A$. Można udowodnić twierdzenie 7.

**Twierdzenie 7** (Kryterium Hermite'a)

Niech $F_q$ będzie ciałem o charakterystyce $p$. Wielomian $g \in F_q$ [10] jest wielomianem permutacyjnym nad $F_q$ wtedy i tylko wtedy, gdy zachodzą następujące związki:

1) $g(X)$ ma dokładnie jeden pierwiastek w $F_q$,
2) dla każdego $k \in \{1, 2, \ldots, q-2\}$, które nie jest wielokrotnością $p$, wielomian

$$[g(X)]^k \pmod{(X^q - X)}$$

ma stopień nie większy niż $q-2$.

Dowód twierdzenia można znaleźć w [10].

***Przykłady wielomianów permutacyjnych***

**Przykład 6**

Każdy wielomian $g$ postaci

$$g(X) = X^{\frac{q+1}{2}} + aX$$

jest wielomianem permutacyjnym, gdy $a$ jest tak dobrane, że symbol Legendre'a $\left(\frac{a^2-1}{q}\right) = 1$, gdzie $q$ jest nieparzystą liczbą pierwszą.

**Przykład 7**

Każdy wielomian $g$ postaci

$$g_k(X, a) = \sum_{j=0}^{[k/2]} \frac{k}{k-j} \left(\frac{k-j}{j}\right) (-a)^j X^{k-2j}$$

jest dla $a \in F_q^*$ wielomianem permutacyjnym nad $F_q$ wtedy i tylko wtedy, gdy największy wspólny dzielnik $(k, q^2-1) = 1$. Wielomiany te nazywa się w literaturze [10] wielomianami Dicksona.

**Przykład 8**

Każdy jednomian $X^n$ jest wielomianem permutacyjnym nad $F_q$ wtedy i tylko wtedy, gdy $(n, q-1) = 1$.

Oczywiście wykorzystanie wielomianów permutacyjnych ma praktyczny sens wówczas, gdy są one niskiego stopnia lub nie mają zbyt wielu niezerowych współczynników. W przeciwnym przypadku znajdowanie wartości wielomianu permutacyjnego jest kłopotliwe czasowo. Inne przykłady wielomianów permutacyjnych można znaleźć w monografii [10].

## *Generatory inwersyjne*

Generatory inwersyjne pojawiły się jako efekt poszukiwań takich przekształceń, które generują ciąg wyrazów (liczb) nie cechujący się lokalną liniowością [2 ÷ 4, 7]. Generatory te są zadane przez następującą formułę:

$$X_{n+1} = \begin{cases} aX_n^{-1} + b \pmod p, & X_n \geqslant 1 \\ b, \text{ dla } X_n = 0 \end{cases} \tag{10}$$

dla $n = 0, 1, \ldots$ .

Podstawową własność generatora inwersyjnego, dotyczącą długości okresu wyraża twierdzenie 8.

**Twierdzenie 8**

Niech $X_0 = 0$ oraz $\lambda = \min\{k \geqslant 1 : X_k = X_0\}$ oznacza długość okresu generatora inwersyjnego, opisanego za pomocą wzoru (10). Liczba $\lambda$ spełnia następujące warunki:

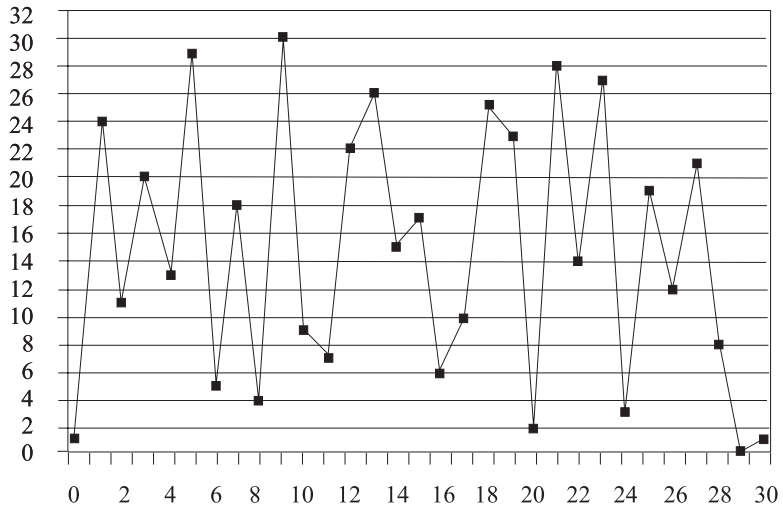
1) $\lambda = p-1$, jeśli $4a + b^2 \equiv 0 \pmod p$;
2) $\lambda + 1$ dzieli $p-1$, jeśli $4a + b^2$ nie dzieli się przez $p$ i jest resztą kwadratową modulo $p$;
3) $\lambda + 1$ dzieli $p+1$, jeśli $4a + b^2$ nie dzieli się przez $p$ i jest niereszta kwadratową modulo $p$.

Twierdzenie to zaczerpnięto z pracy [4].

**Przykład 9**

Generator (10) dla $a = 23$, $b = 1$ wytwarza następującą sekwencję:
0, 1, 24, 11, 20, 13, 29, 5, 18, 4, 30, 9, 7, 22, 26, 15, 17, 6, 10, 25, 23, 2, 28, 14, 27, 3, 19, 12, 21, 8, 0, ...

Zależność funkcyjną następnego wyrazu generatora inwersyjnego w funkcji poprzedniego przy powyższych parametrach przedstawiono na rys. 9.




***Rys. 9.*** *Wykres zależności generatora inwersyjnego (10) w funkcji wyrazu poprzedniego* $X_{n+1} = f(X_n)$ *dla parametrów* $a = 23$, $b = 1$

Ponieważ $4a + b^2 = 93$ dzieli się przez 31, więc okres $\lambda$ rozpatrywanego generatora – zgodnie z twierdzeniem 8 – jest równy $31 - 1 = 30$.

Generator (10) dla $a = 22$, $b = 1$ wytwarza następującą sekwencję:
0, 1, 23, 6, 15, 19, 25, 18, 16, 14, 7, 13, 17, 26, 9, 0, ...

Zależność funkcyjną następnego wyrazu generatora inwersyjnego w funkcji poprzedniego przy powyższych parametrach zaprezentowano na rys. 10.

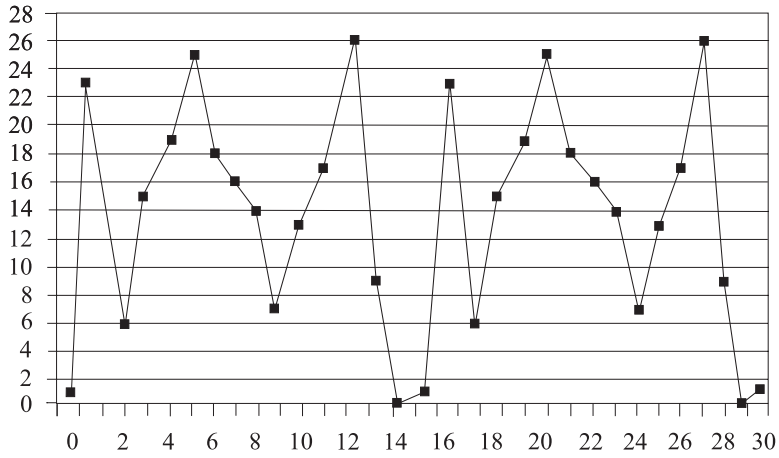
Ponieważ $4a + b^2 = 89$ nie dzieli się przez 31 oraz 96 jest niereszą kwadratową modulo 31, więc okres $\lambda = 15$ rozpatrywanego generatora – zgodnie z twierdzeniem 8 – spełnia warunek $\lambda + 1 \mid p + 1 = 32$.

Generator (10) dla $a = 24$, $b = 1$ wytwarza następującą sekwencję:
0, 1, 25, 28, 24, 2, 13, 10, 22, 19, 30, 8, 4, 7, 0, ...

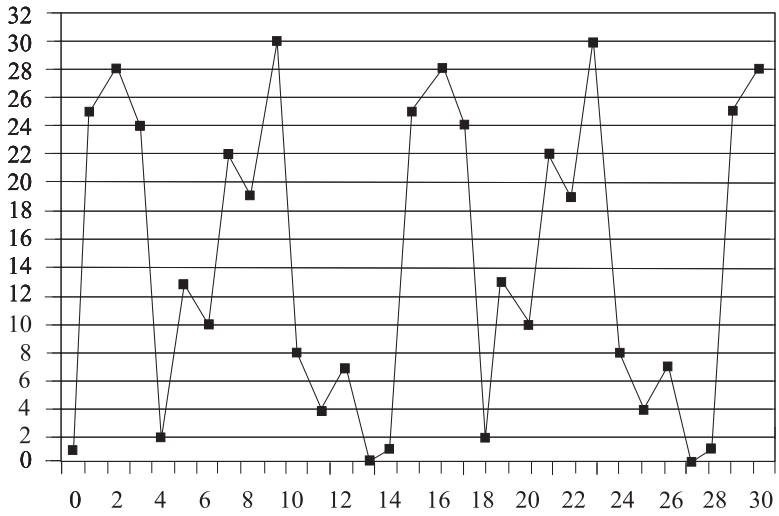
Zależność funkcyjną następnego wyrazu generatora inwersyjnego w funkcji poprzedniego przy powyższych parametrach pokazano na rys. 11.

Ponieważ $4a + b^2 = 97$ nie dzieli się przez 31 oraz 96 jest resztą kwadratową modulo 31, więc okres $\lambda = 14$ rozpatrywanego generatora – zgodnie z twierdzeniem 8 – spełnia warunek $\lambda + 1 \mid p - 1 = 30$.

Z punktu widzenia zastosowań jest ważny przypadek, gdy moduł generatora inwersyjnego stanowi potęgę liczby 2. Można wówczas wykazać, że dla liczby modularnej $2^m$ dla $m > 2$ i warunku




***Rys. 10.*** *Wykres zależności generatora inwersyjnego (10) w funkcji wyrazu poprzedniego* $X_{n+1} = f(X_n)$ *dla parametrów* $a = 22,\ b = 1$




***Rys. 11.*** *Wykres zależności generatora inwersyjnego (10) w funkcji wyrazu poprzedniego* $X_{n+1} = f(X_n)$ *dla parametrów* $a = 24,\ b = 1$

początkowego $X_0 = 1$, maksymalny okres generatora inwersyjnego jest równy $2^{m-1}$ dokładnie wtedy, gdy liczba $a$, występująca w definicji (10) generatora inwersyjnego, daje w wyniku dzielenia przez 4 resztę 1, liczba $b$ zaś resztę 2 [7]. Można udowodnić, że kolejne wyrazy generatorów inwersyjnych nie mają własności lokalnej liniowości [2], tzn. co najwyżej dwa punkty $(X_{n-1}, X_n)$, $(X_n, X_{n+1})$ leżą na prostej. Co najwyżej trzy kolejne wyrazy $(X_{n-2}, X_{n-1})$, $(X_{n-1}, X_n)$, $(X_n, X_{n+1})$ mogą leżeć w jednej płaszczyźnie itd. Własność ta może wskazywać, że generatory inwersyjne będą trudniej przewidywalne od liniowych i afinicznych. Zasadniczą trudnością, związaną z wykorzystaniem generatorów afinicznych, jest duża złożoność operacji brania odwrotności multyplikatywnej liczby

naturalnej. Szybki algorytm znajdowania inwersji $x^{-1}$ liczby $x$, opierający się na algorytmie Euklidesa, podał Aigner w [1]. Algorytm ten jest następujący.

---

**Algorytm wyznaczania multyplikatywnej odwrotności liczby $x$**

Krok 1: Przyjmuje się $z_{n+2} = p$, $z_{n+1} = x$ i wyznacza monotonicznie malejący ciąg dodatnich liczb całkowitych $z_n, \ldots, z_2, z_1 = 1$ o tej własności, że $z_i = z_{i+2} - q_i \cdot z_{i+1}$, $1 \leqslant i \leqslant n$.

Krok 2: Tworzy się ciąg $(w_i)_{i=1,2,\ldots,n-2}$, zgodnie z następującymi zależnościami: $w_1 = 0$, $w_2 = 1$ oraz $w_i = q_{i-1} \cdot w_{i-1} + w_{i-2}$, $3 \leqslant i \leqslant n+2$. Multyplikatywna odwrotność liczby $x$ jest dana w następującej postaci: $x^{-1} = (-1)^{n+2} \cdot w_{n+2}$.

---

**Przykład 10**

Można przyjąć, że $p = 31$, $x = 7$. Na podstawie algorytmu Aignera można znaleźć multyplikatywną odwrotność liczby 7 modulo 31.

$$\begin{aligned} 31 - 4 \cdot 7 = 3, &\qquad (z_4 - q_3 \cdot z_3 = z_2), \\ 7 - 2 \cdot 3 - 1, &\qquad (z_3 - q_2 \cdot z_2 = z_1). \end{aligned}$$

Otrzymuje się zatem $z_4 = 31$, $z_3 = 7$, $z_2 = 3$, $z_1 = 1$ oraz $q_2 = 2$, $q_3 = 4$. Zgodnie z algorytmem Aignera, buduje się ciąg $(w_i)_{i=1,4}$. Jest $w_1 = 0$, $w_2 = 1$,

$$\begin{aligned} w_3 &= q_2 \cdot w_2 + w_1 = 2 \cdot 1 + 0 = 2, \\ w_4 &= q_3 \cdot w_3 + w_2 = 4 \cdot 2 + 1 = 9, \end{aligned}$$

zatem

$$7^{-1} = (-1)^4 \cdot 9 = 9.$$

Z łatwością można sprawdzić, że $7 \cdot 9 = 63 \equiv 1 \pmod{31}$.

Przedstawiony algorytm wymaga średnio $\left(\frac{12}{\pi^2} \ln 2\right) \cdot \ln p$ kroków [4]. Wyznaczenie kolejnej pseudopierwszej liczby za pomocą generatora inwersyjnego (10) trwa – średnio rzecz biorąc – kilkanaście razy dłużej niż z wykorzystaniem generatora liniowego czy afinicznego dla liczb kilkudziesięciocyfrowych. W przypadku liczb kilkusetcyfrowych generator inwersyjny działa kilkadziesiąt razy wolniej niż wcześniej wymienione generatory liniowe i afiniczne. Dlatego też generatory inwersyjne warto stosować jedynie w uzasadnionej potrzebie, np. jeżeli chce się uniknąć lokalnej liniowości typowej dla generatorów liniowych i afinicznych. Generator inwersyjny może być efektywnie realizowany sprzętowo. Można wtedy wykorzystać algorytm liczenia multyplikatywnej odwrotności przez potęgowanie liczby.

## *Bibliografia*

[1] Aigner A.: *Zahlentheorie*. Berlin-NewYork, de Gruyter Verl., 1975

[2] Eichenauer-Herrmann J.: *Inversive congruential pseudorandom numbers avoid the planes*. Mathematics of Computation, 1991, vol. 56, no. 193, s. 297–301

[3] Eichenauer-Herrmann J., Tpuzoglu A.: *On the period length of congruential pseudorandom number sequences generated by inversions*. Journal of Computational and Applied Math., 1990, vol. 31, s. 87–96

[4] Eichenauer J., Lehn J.: *A non-linear congruential pseudo random number generator*. Statistische Hefte, 1986, nr 27, s. 315–326

[5] Elliott P.D.T.A., Murata L.: *On the average of the least primitive root modulo p*. J. London Math. Soc., 1997, vol. 2, no. 56, s. 435–454

[6] Hooley C.: *On artin's conjecture*. J. f. Reine und Angewendte Mathematik, 1967, Bd. 225, s. 209–220

[7] Kato T., Li-Ming Wu, Yanagihara N.: *On a nonlinear congruential pseudorandom number generator*. Mathematics of Computation, 1996, vol. 65, no. 213, s. 227–233

[8] Knuth D.E.: *The Art of Computer Programming*. Vol. 2, Seminumerical Algorithms, 3rd ed. Reading Massachusetts, Addison-Wesley, 1997

[9] Landau E.: *Vorlesungen über Zahlentheorie*. Leipzig, Dritter Band, Hirzel, 1927

[10] Lidl R., Niederreiter H.: *Finite Fields*. Addison-Wesley Publ. Company, 1983

[11] Paszkiewicz A.: *Własności asymptotyczne najmniejszych pierwiastków pierwotnych liczb pierwszych*. Materiały z KST'98, Bydgoszcz, 1998, t. B, s. 133–142

[12] Zieliński R.: *Generatory liczb losowych*. Warszawa, WNT, 1972

## *Andrzej Paszkiewicz*

Dr inż. Andrzej Paszkiewicz (1957) – absolwent Wydziału Cybernetyki Wojskowej Akademii Technicznej (1981); pracownik naukowy Wojskowego Instytutu Łączności (1981–1996) oraz Politechniki Warszawskiej (od 1996); autor lub współautor ponad 40 prac naukowych oraz autor kilku zgłoszeń patentowych; zainteresowania naukowe: kryptografia i ochrona informacji, teoria kodowania, grafy losowe, teoria chaosu, niezawodność i bezpieczeństwo sieci komputerowych, certyfikacja urządzeń i algorytmów kryptograficznych.
e-mail: anpa@tele.pw.edu.pl