Metryki losowości, testy FIPS 140-2 generatorów liczb pseudolosowych

Poziom 4

Michał Artur Szlupowicz, Marek Brynda

Marzec, 2020

1 Liniowy generator kongruentny

Liniowy generator kongruentny generuje kolejne wartości na podstawie poniższego wzoru (źródło [1]):

$$x_i = (a \cdot x_{i-1} + c) \mod m \tag{1}$$

Użyte parametry (Borland C/C++) (źródło [2]):

- m = 2^{32}
- a = 22695477
- c = 1

2 Wyniki testów

Dla $x_0 = 5$ otrzymaliśmy następujące wyniki:

```
rngtest: starting FIPS tests...
rngtest: entropy source drained
rngtest: bits received from input: 3144960
rngtest: FIPS 140-2 successes: 24
rngtest: FIPS 140-2 failures: 133
rngtest: FIPS 140-2(2001-10-10) Monobit: 133
rngtest: FIPS 140-2(2001-10-10) Poker: 133
rngtest: FIPS 140-2(2001-10-10) Runs: 133
rngtest: FIPS 140-2(2001-10-10) Long run: 133
rngtest: FIPS 140-2(2001-10-10) Continuous run: 106
rngtest: input channel speed: (min=10000000000.000; avg=21216216216.216; max=0.000)bits/s
rngtest: FIPS tests speed: (min=142.339; avg=339.555; max=489.064)Mibits/s
rngtest: Program run time: 9001 microseconds
```

Nasz plik nie przeszedł w około 15% przypadków testów (źródła [1] [3]):

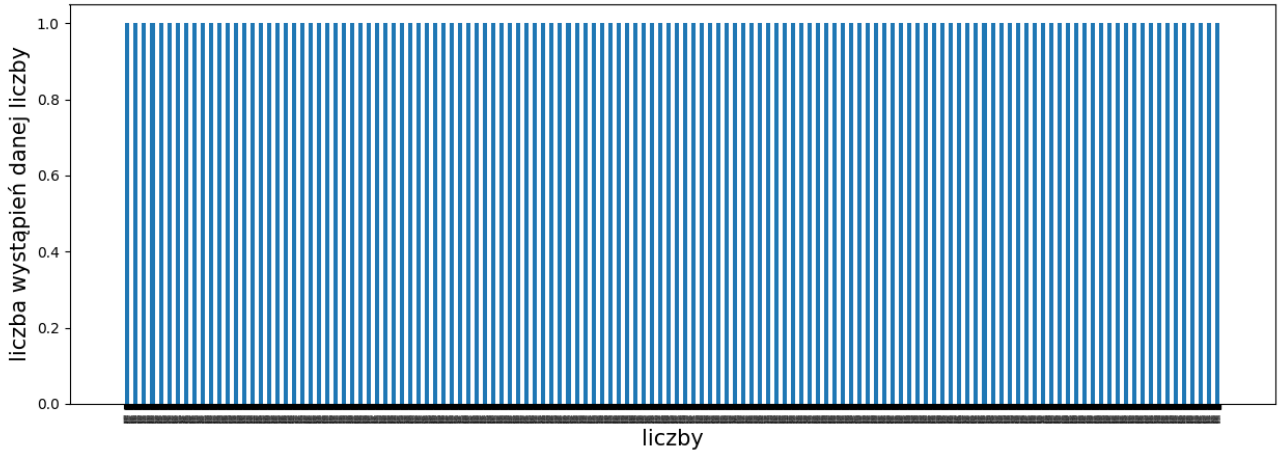
1. Monobit (w każdym pakiecie po 20000 bitów było o 275 więcej 0 lub jedynek),

2. Poker (w każdym 4 bicie zostaje zliczona liczba wystąpień wartości z przedziału 0 - 0xf., podnosi je do kwadratu w momencie gdy któraś wartość nie mieści się w przedziale $< 1563176; 1576928 >$ zwraca fałsz),

3. Runs (zliczanie liczby ciągów długości od 1 do co najmniej 6 składających się z samych jedynek lub zer oraz porównaniu z przedziałami z tabeli
   1-bit: $2315 <$ zliczeń < 2685
   2-bit: $1114 <$ zliczeń < 1386
   3-bit: $527 <$ zliczeń < 723
   4-bit: $240 <$ zliczeń < 384
   5-bit: $103 <$ zliczeń < 209
   6-bit (6 i więcej bitów): $103 <$ zliczeń < 209)

4. Long run (zaobserwowanie 26 lub większej liczby bitów takich samych obok siebie ('0' lub '1'))

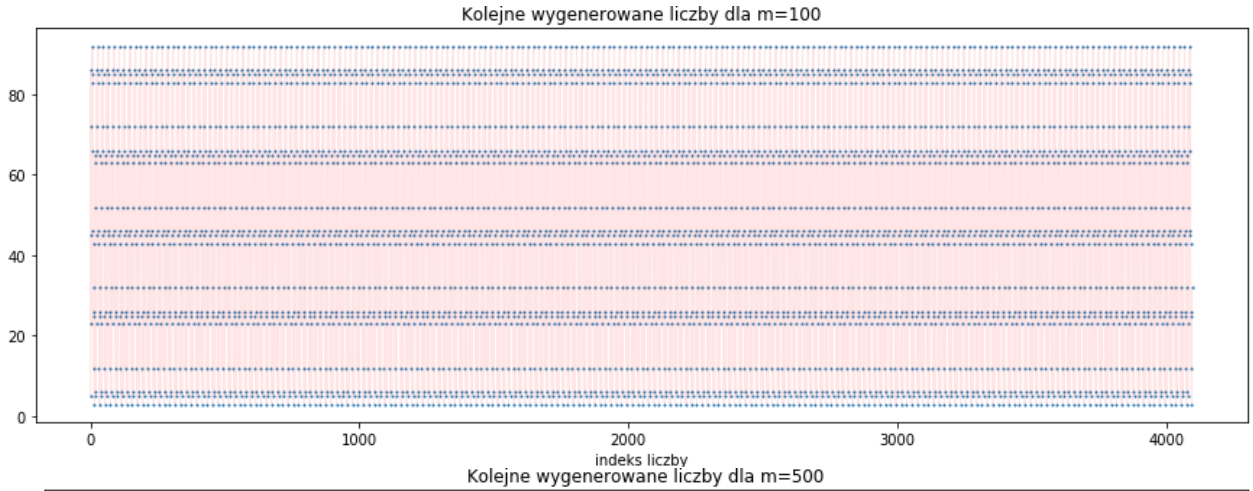
5. Continous run (zaobserwowanie dwa raz tego samego wzoru o długości 32 bitów obok siebie)

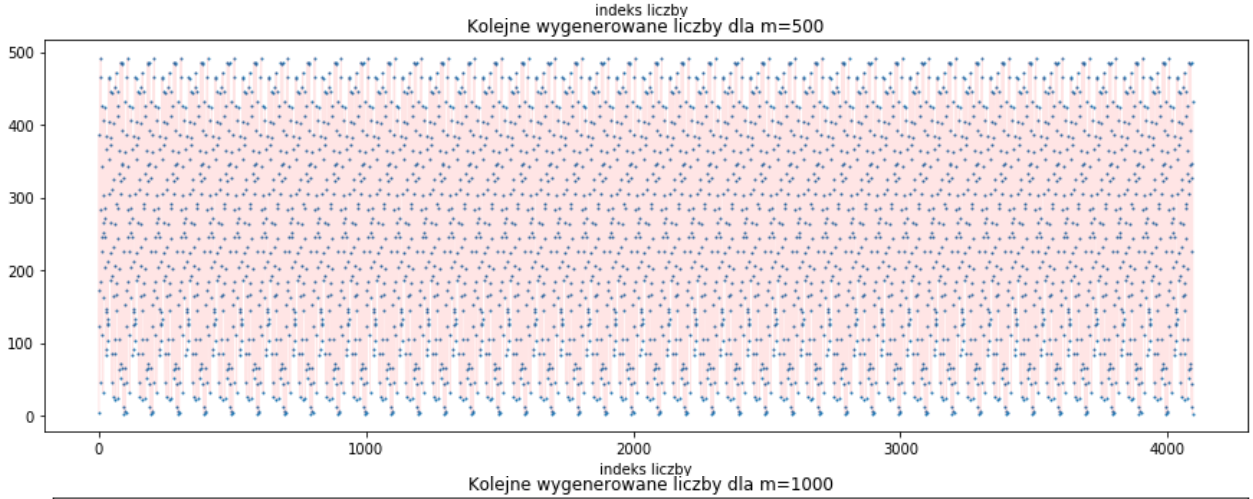


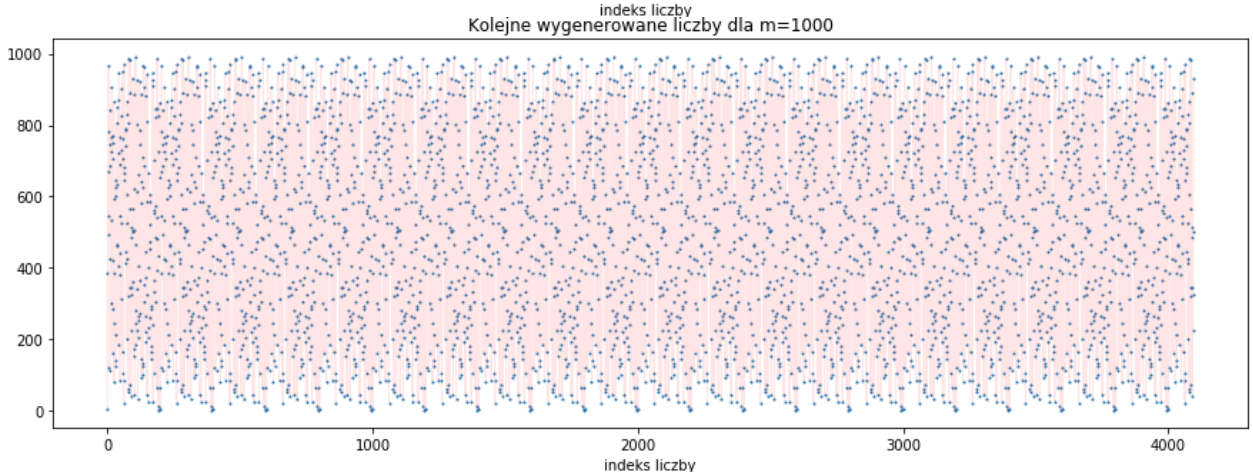
Rysunek 1: Histogram liczba wystąpień każdej z wygenerowanych liczb

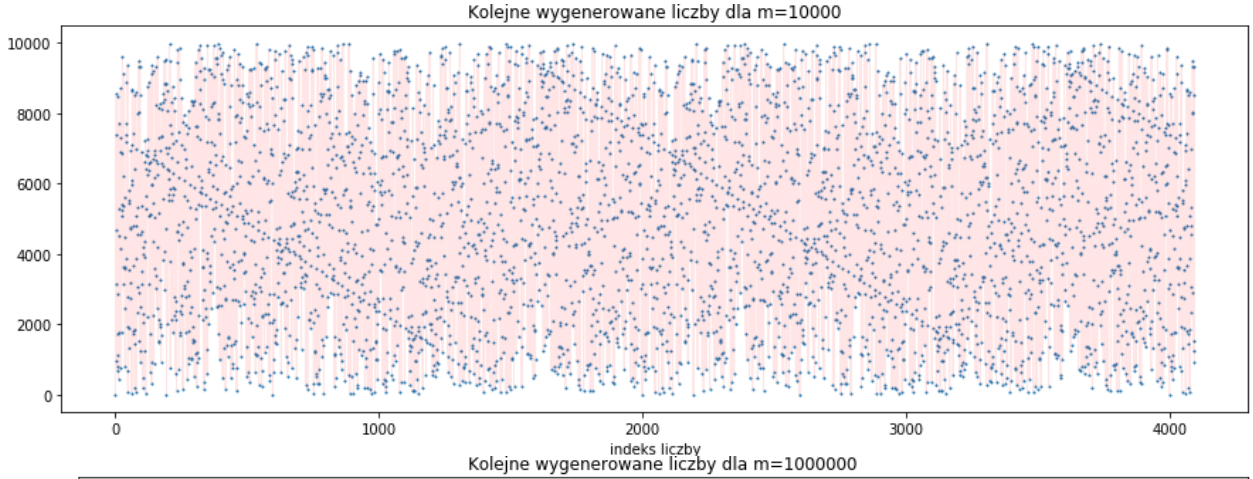
3 Wzorce

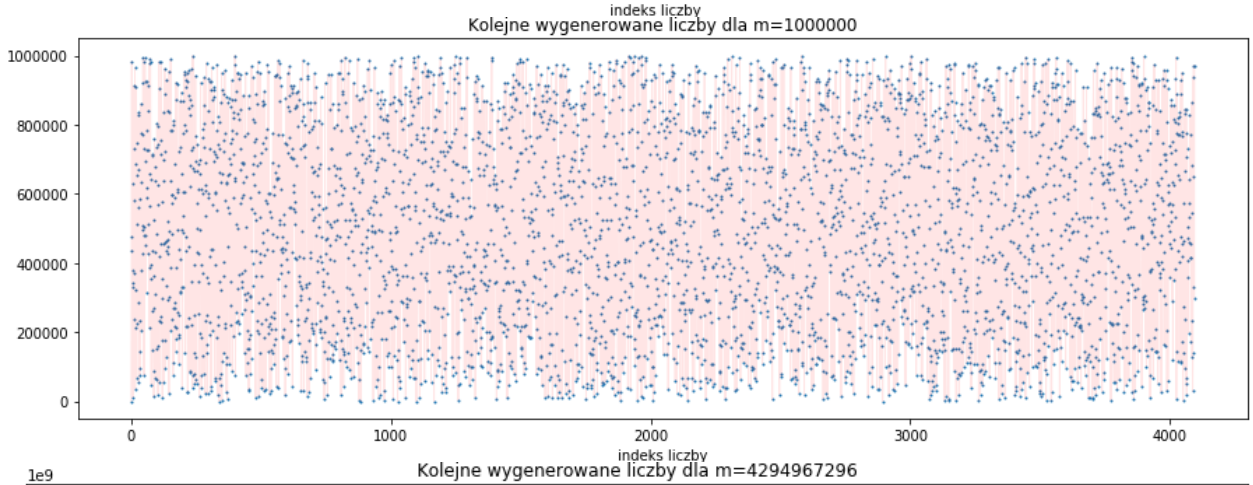
Zainspirowani [4] postanowiliśmy sprawdzić wygenerowany przez nas ciąg dla parametru $m \in \{100, 500, 1000, 10^4, 10^6, 2^{32}\}$.

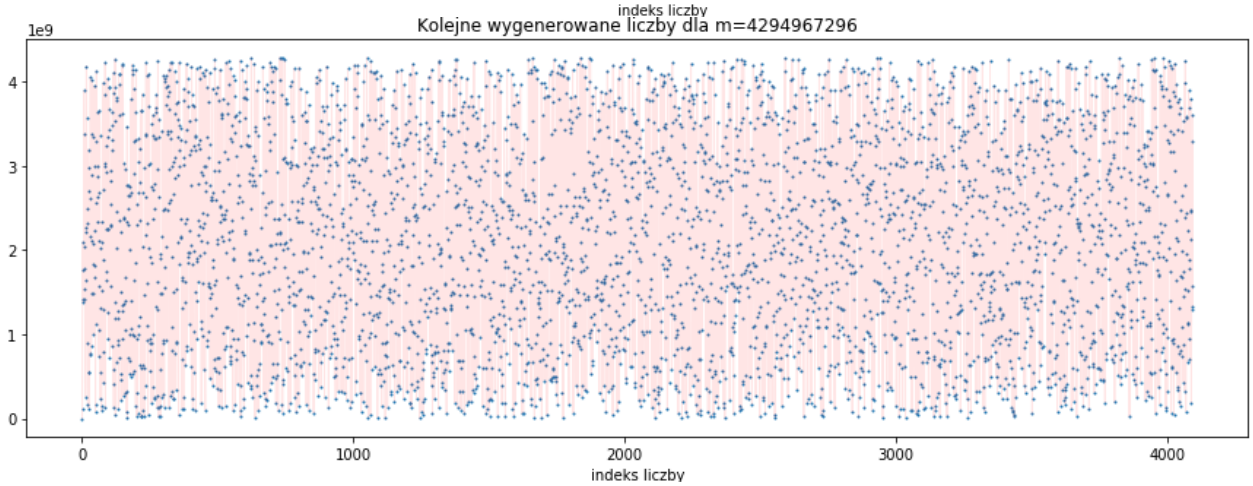


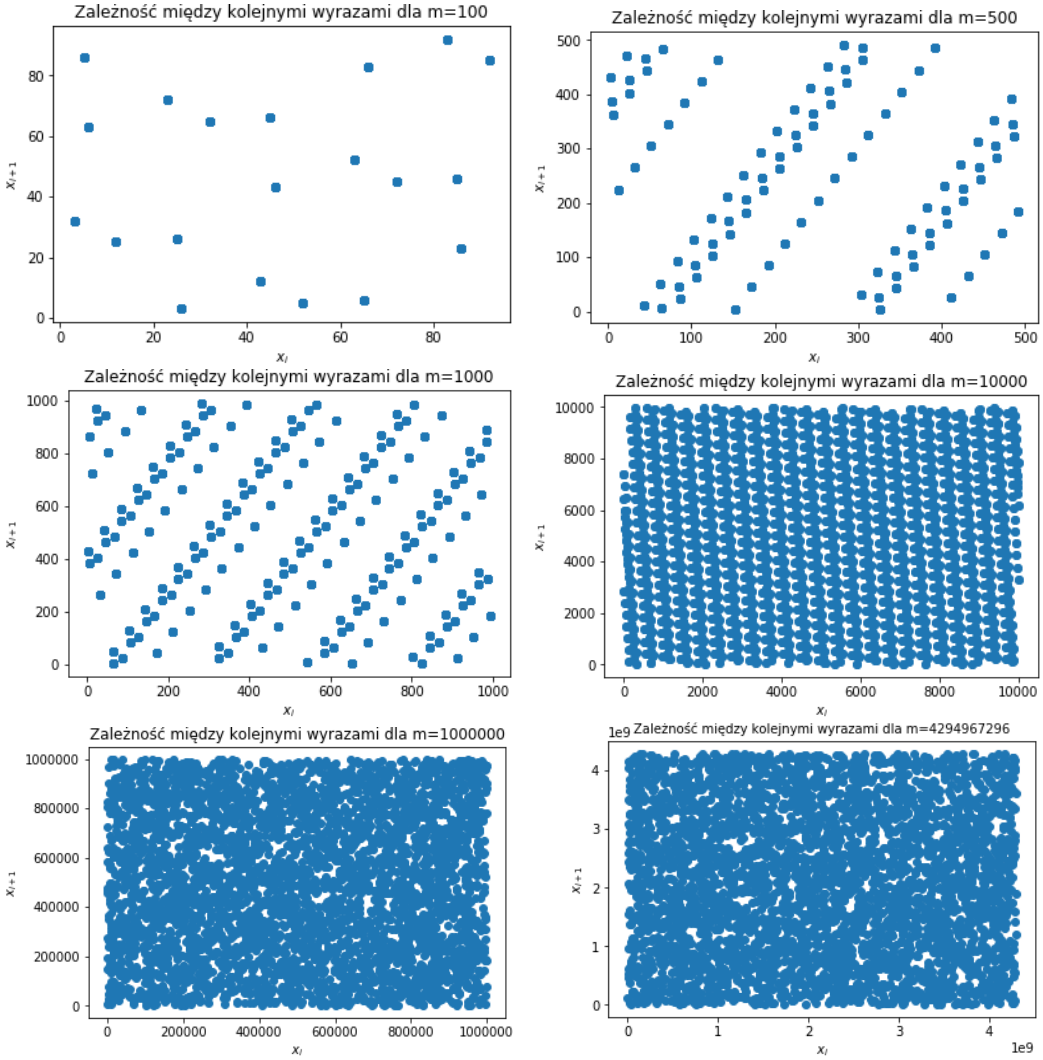





Kolejne wygenerowane liczby dla m=10000
10000
8000
6000
4000
2000
0
0
1000
2000
3000
4000
indeks liczby


Kolejne wygenerowane liczby dla m=1000000
1000000
800000
600000
400000
200000
0
0
1000
2000
3000
4000
indeks liczby


Kolejne wygenerowane liczby dla m=4294967296
1e9
4
3
2
1
0
0
1000
2000
3000
4000
indeks liczby



Rysunek 2: Wykresy przedstawiające zależność miedzy kolejnymi wyrazami ciągu

4 Podsumowanie

Rysunek 1 wygląda tak jakbyśmy oczekiwali - żadna liczna nie powtarza się z większą częstotliwością niż pozostałe.
Natomiast łatwo można zauważyć powtarzające się wzorce dla mniejszych wartości parametru m, gdy przyjrzymy się kolejnym wygenerowanym liczbom, widać też wyraźnie zależność pomiędzy kolejnymi wyrazami ciągu x_i oraz x_{i+1}.
Ciąg znaków przeszedł w niektórych przypadkach testy, co pokrywa się z

wynikami z publikacji [1], pokazującymi że przy wysokich wartościach parametrów użytych w generatorze możliwe jest osiągnięcie pseudolosowych ciągów przechodzących testy FIPS 140-2.

Literatura

[1] Kadir, Rashidah & Maarof, Mohd. (2009). A Comparative Statistical Analysis of Pseudorandom Bit Sequences. 5th International Conference on Information Assurance and Security, IAS 2009. 2. 91-94. 10.1109/IAS.2009.242.

[2] Tomasz Lubiński, Generator LCG (Liniowy Generator Kongruentny) http://www.algorytm.org/liczby-pseudolosowe/generator-lcg-liniowy-generator-kongruentny.html

[3] Whirlygig Verification and rngtest analysis https://warmcat.com/2009/05/21/whirlygig-verification-and-rngtest-analysis.html

[4] Cheng Sun, Linear congruential generator visualisation https://chengsun.uk/lcg.html