Metody Probabilistyczne i Statystyka

Zadanie domowe 3

Michał Łukomski

255552

```
begin
    # biblioteki używane w tym ćwiczeniu
    using Random
    using SHA
    using Plots
    using StatsPlots
    using Distributions
    using LaTeXStrings
    using StableRNGs
    using PlutoUI
end
```

Zad 1 - testy NIST

bits_length = 10000000

```
bits_length = 10_000_000
```

seed = 21

```
seed = 21 # for reproducibility, note that RandomDevice, always produce different results
```

To see the output, uncomment `#|> println`

linear congruential generator (LCG)

"11101110111011011001010010110100001100110000111100000010111000111001100011110101011011100

```
rand(StableRNG(seed), Bool, bits_length) .|>
UInt8 .|> string |> join #|> println
```

MersenneTwister - requires much more space to store the state ector and generate a random sequence than Xoshiro256++

```
"000100011011000100110001000000010101111000000000000011101101000110010001000110000011101 1
```

```julia
rand(MersenneTwister(seed), Bool, bits_length) .|>
UInt8 .|> string |> join # |> println
```

Xoshiro256++ - default Julia algorithm

```
"10101011100010001100100001011110110101101110110111111110001110100000101111001111010100001
```

```julia
rand(Xoshiro(seed), Bool, bits_length) .|>
UInt8 .|> string |> join # |> println
```

RandomDevice - Entropy obtained from the operating system - the only cryptographically secure random number generator from the ones tested, always produce different results

```
"0011111011010001010011000001010000101010010111110100101000111101011000010110011001000001
```

```julia
rand(RandomDevice(), Bool, bits_length) .|>
UInt8 .|> string |> join  #|> println
```

Output zwrócony przez funkcję hashującą SHA-1 dla własnego nazwiska

```julia
sha1("Łukomski") .|> bitstring |> join |> println
```

```
110100010110011000011100111101011111110010000010111010001100110011011001
01100111110011110110101100011001010000011100101011010100010111001110110000001
000001011110
```

Wyniki testów NIST zostały przedstawione w poniższej tabeli, dokładne zrzuty ekranu z otrzymanymi p-value znajdują się na końcu dokumentu. Generatory testowano na sekwencji 10 milionów bitów każdy. Z testowanych generatorów trzy, tj. `LCG`, `Xoshiro` oraz fizyczny rng komputera, przeszły wszystkie testy, a `MersenneTwister` przeszedł prawie wszystkie testy, poza *Random Excursion Test*. Output funkcji SHA-1 dla nazwiska autora - "Łukomski" - był za krótki dla 8 z 14 testów, jednak przeszedł pomyślnie wszystkie testy dla których miał odpowiednią długość.

Test name	LCG	MersnneTwister	Xoshiro	RandomDevice	SHA1
1. Frequency (Monobit) Test	Passed	Passed	Passed	Passed	Passed
2. Frequency Test within a Block	Passed	Passed	Passed	Passed	Passed
3. Runs Test	Passed	Passed	Passed	Passed	Passed
4. Test for the Longest Run of Ones in a Block	Passed	Passed	Passed	Passed	Passed
5. Binary Matrix Rank Test	Passed	Passed	Passed	Passed	
6. Non-overlapping Template Matching Test	Passed	Passed	Passed	Passed	
7. Overlapping Template Matching Test	Passed	Passed	Passed	Passed	
8. Maurer's "Universal Statistical" Test	Passed	Passed	Passed	Passed	
9. Linear Complexity Test	Passed	Passed	Passed	Passed	
10. Serial Test	Passed	Passed	Passed	Passed	
11. Approximate Entropy Test	Passed	Passed	Passed	Passed	Passed
12. Cumulative Sums (Cusum) Test	Passed	Passed	Passed	Passed	Passed
13. Random Excursions Test	Passed	**Failed**	Passed	Passed	
14. Random Excursions Variant Test	Passed	Passed	Passed	Passed	

Wnioskuje się więc, że wszystkie badane generatory liczb pseudolosowych mają poprawne właściwości i z powodzeniem przybliżają losowanie liczb losowych. Do kryptograficznych zastosowań jednak, zaleca się używanie jedynie `RandomDevice`. Należy wziąć pod uwagę też, że testy te były przeprowadzane tylko dla jednego ziarna (`seed=21` w tym przypadku) i mogą się różnić dla innych ziaren, co przy dokładniejszym badaniu generatorów trzeba by wziąć pod uwagę i zbadać wyniki dla wielu ziaren.

Zad 2 - Błądzenie losowe na liczbach całkowitych

Zmienna losowa $S_N = \sum_{n=1}^{N} X_n$, gdzie X_n są niezależne i każda przyjmuje wartości 1 oraz -1 z prawdopodobieństwem 0.5. Dla $N = 0$ przyjmuje się $S_0 = 0$.

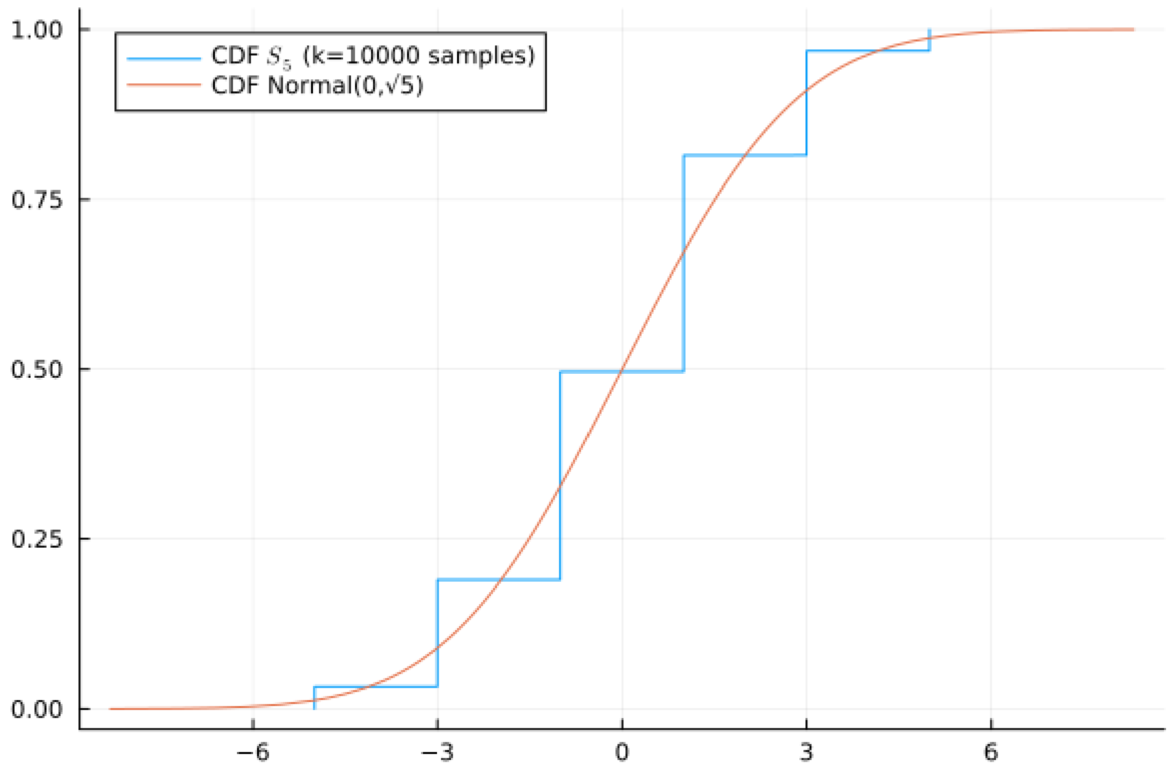
```
S (generic function with 1 method)
```

```
S(N) = rand([-1,1], N) |> sum
```

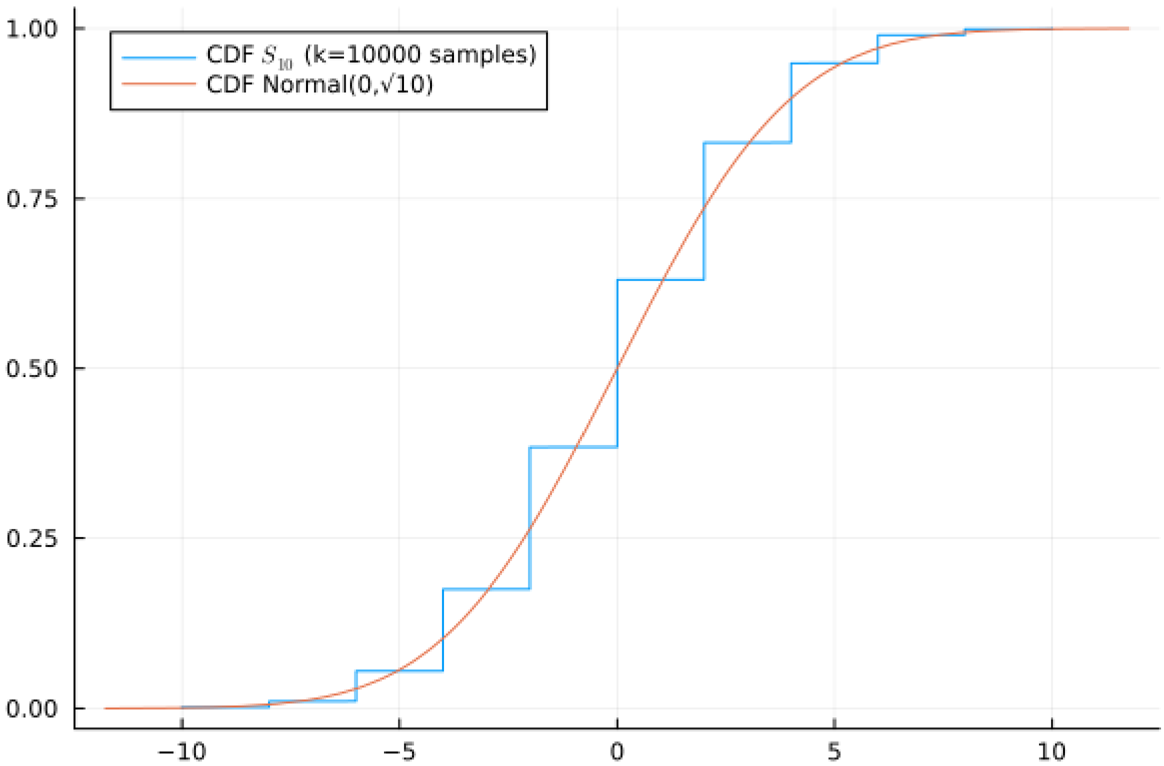
```
drunkard_walk (generic function with 1 method)
```

```
function drunkard_walk(N,k)
    res = [S(N) for _ in 1:k]
    fig = plot(sort(res), (1:k)./k, label=L"CDF $S_{%$N}$ (k=%$k samples)")
    plot!(fig, Normal(0,√N), func=cdf, label="CDF Normal(0,√$N)")
    return fig
end
```

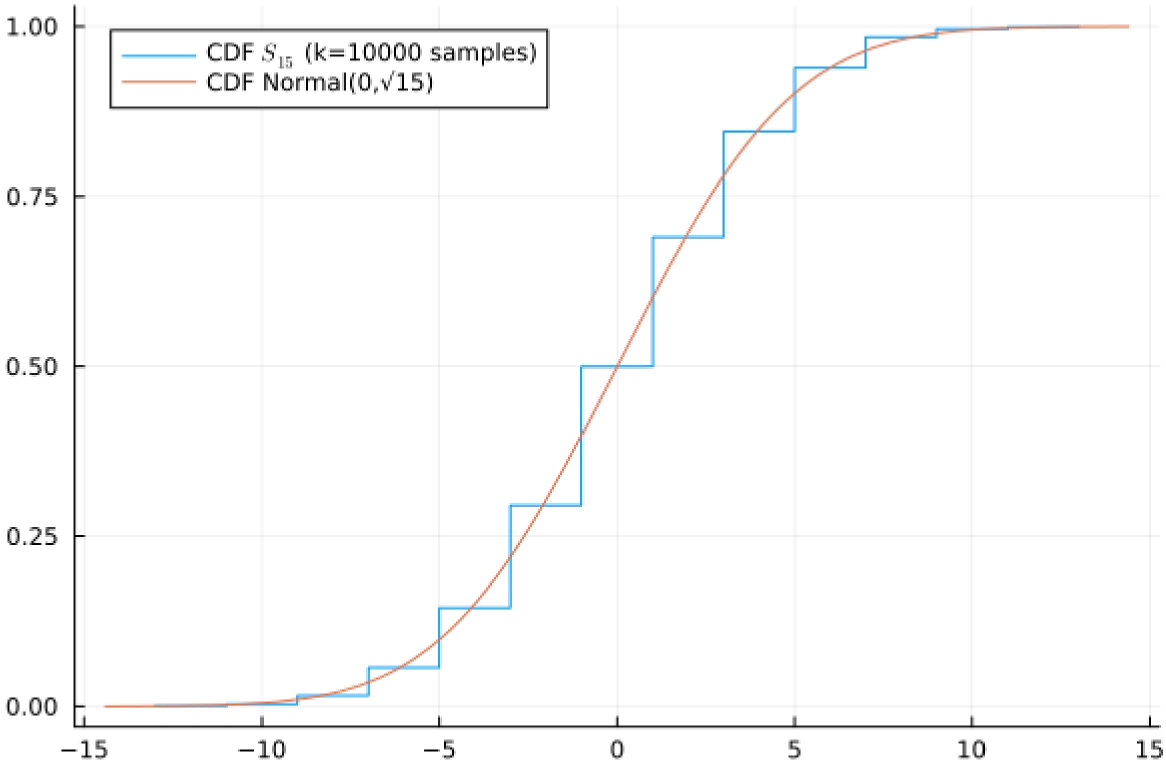
Poniżej przedstawiono wykresy dystrybuanty zmiennej losowej S_N dla $N \in \{5, 10, 15, 20, 25, 30, 100\}$ oraz rozkładu normalnego $\mathcal{N}(\mu = 0, \sigma = \sqrt{N})$, który miałyby aproksymować S_N.



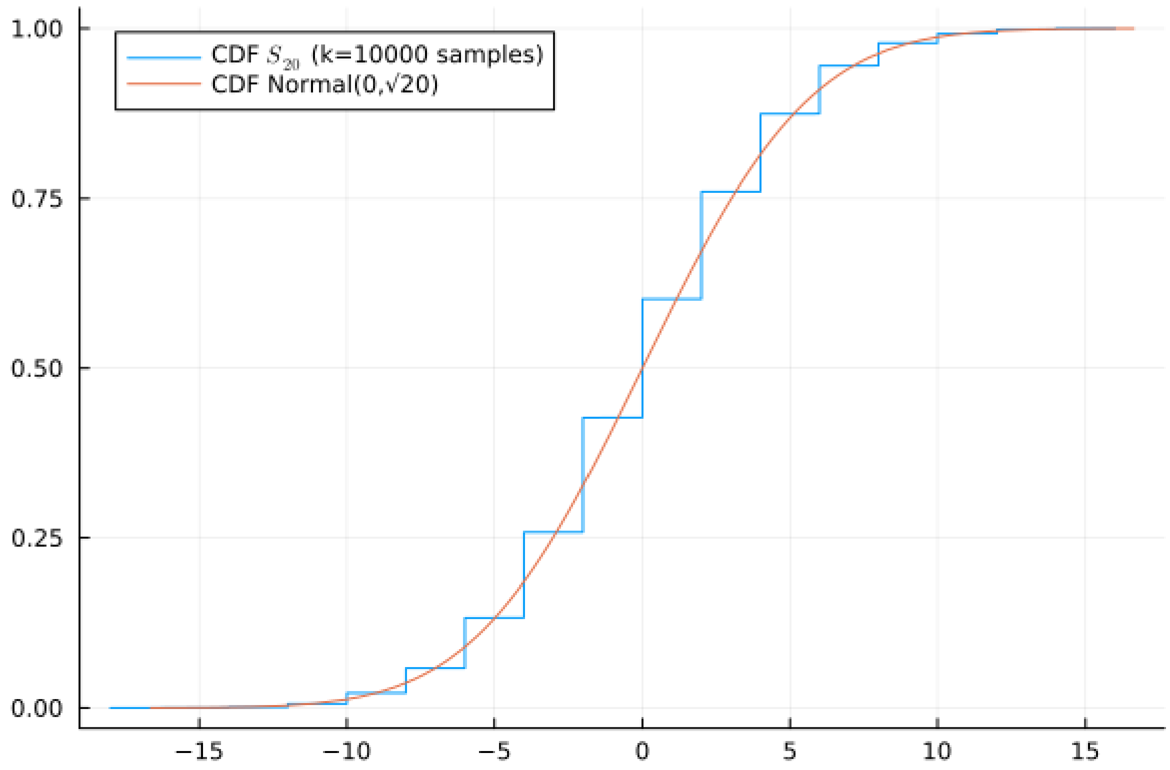

```
drunkard_walk(5,10_000)
```



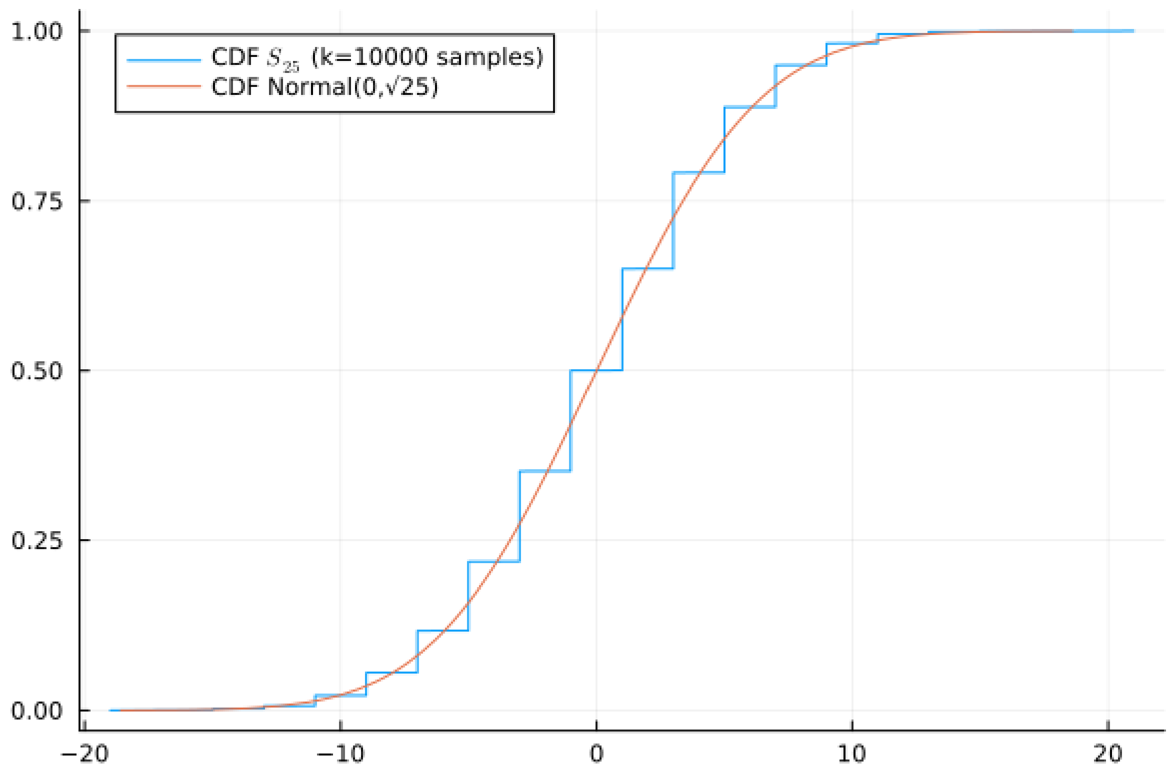

```
drunkard_walk(10,10_000)
```



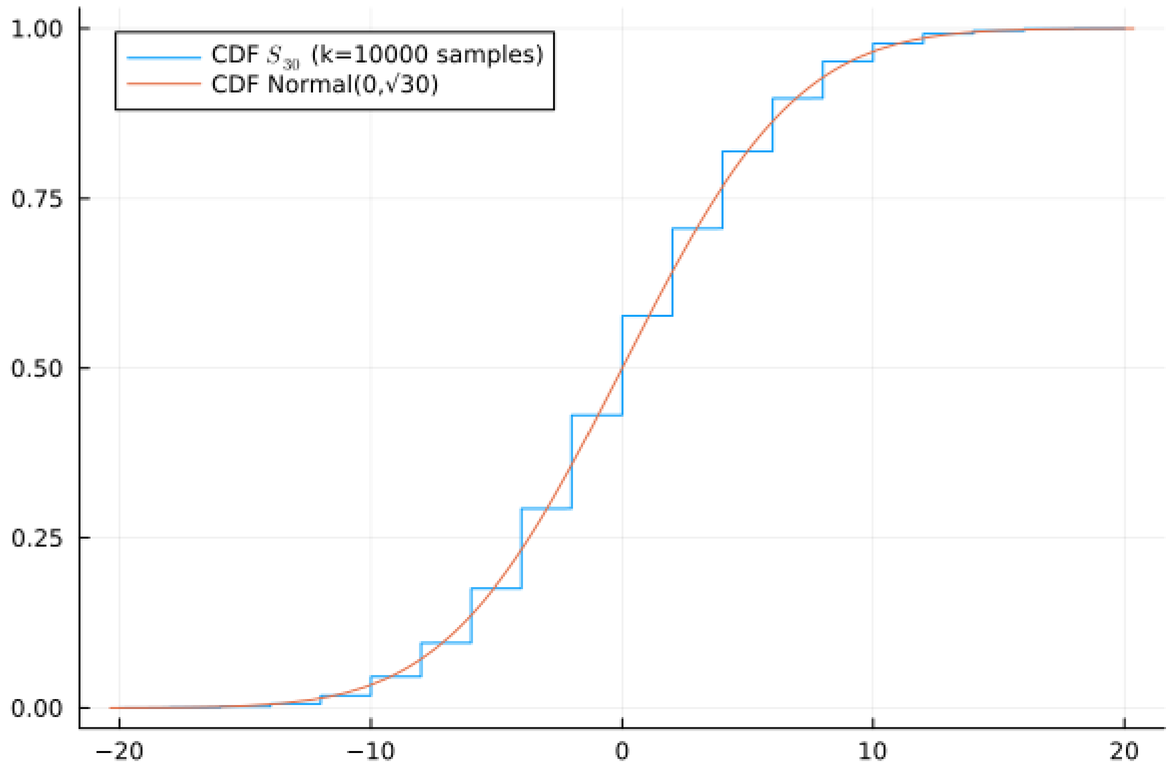

```
drunkard_walk(15,10_000)
```



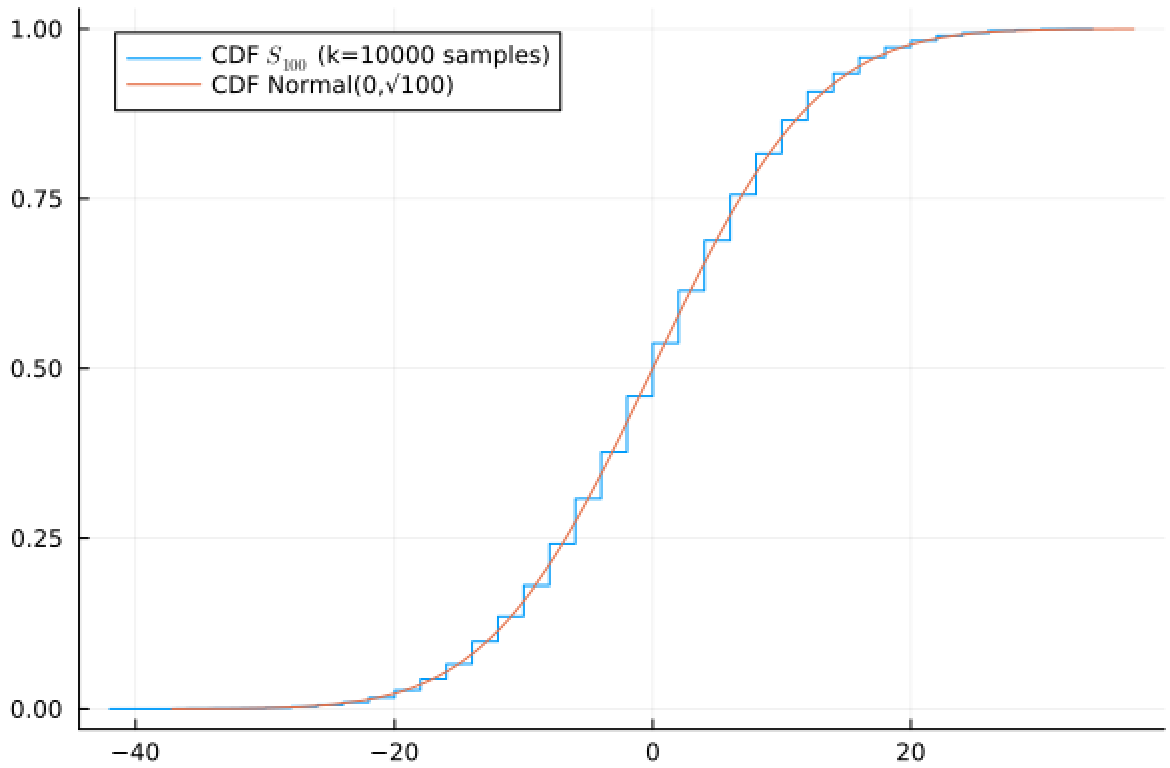

```
drunkard_walk(20,10_000)
```




```
drunkard_walk(25,10_000)
```




```
drunkard_walk(30,10_000)
```




```
drunkard_walk(100,10_000)
```

Można zauważyć, że im większe N, tym bardziej S_N zbliża się do rozkładu normalnego, aczkolwiek nie będą one nigdy sobie równe z uwagi na fakt, że rozkład normalny jest rozkładem ciąłym, a S_N dyskretnym.

Wynika z tego fakt, że dysponując jedynie generatorem losowych, pojedynczych bitów, możeby w przybliżeniu otrzymać generator liczb z rozkładu normalnego - wystarczy generować wystarczająco długi ciąg bitów (wystarczająco duże N), tak aby otrzymać satysfakcjonujące nas przybliżenie, a później przeskalować wynik na docelową średnią i wariancję.

Zad 3 Błądzenie losowe na liczbach całkowitych - rozkład czasu nad osią OX

P_N odpowiada "frakcji czasu" jaki rozważany proces (S_N) "spęda nad osią OX".

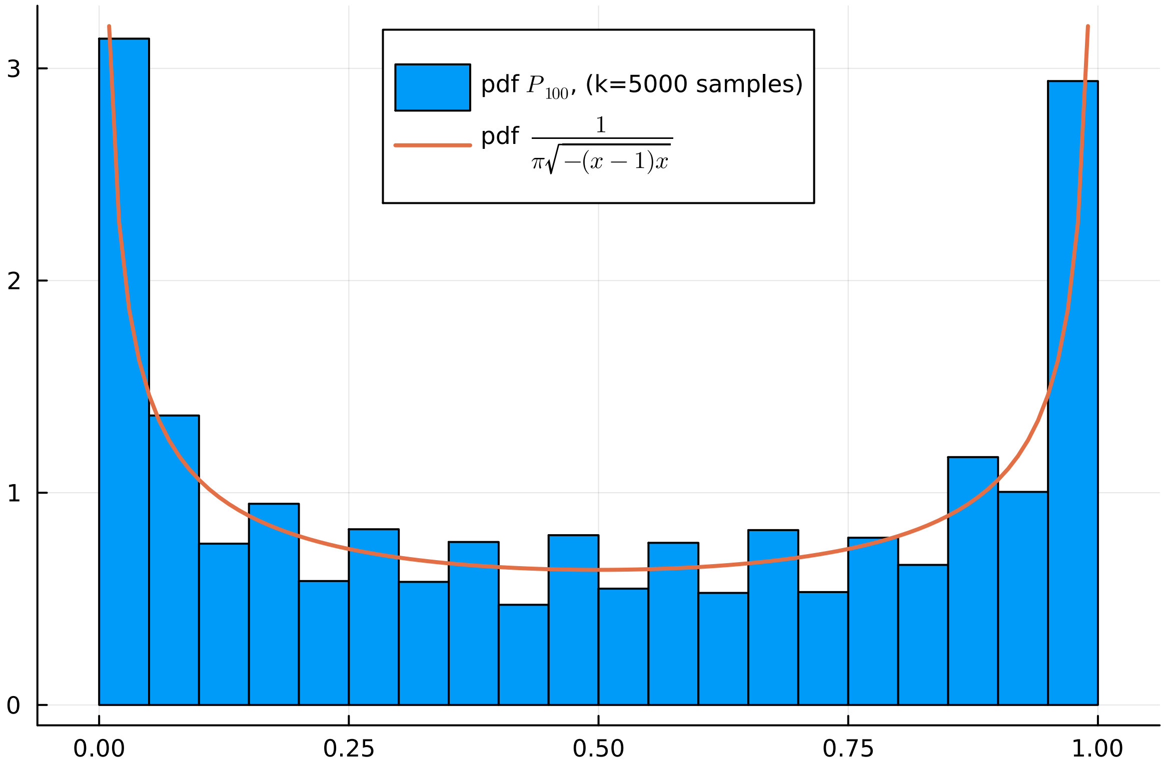
```
frac_over_OX (generic function with 2 methods)
```

```
function frac_over_OX(N,k, nbins=20)
    SN = hcat(zeros(k,1), rand([-1,1], (k,N)))

    cumSN = cumsum(SN, dims=2)

    D_n = reduce(hcat,[cumSN[:,i].>0 .|| cumSN[:,i-1].>0 for i in 2:(N+1)]) # S_0 = 0

    P_N = sum(D_n, dims=2) / N

    # przedziały są [a,b), więc bez nextfloat, 1.0 zostają ucinane
    plt = histogram(
        P_N,
        bins=range(0, nextfloat(1.0), length=nbins+1), # +1, because its the edges
        normalize=:pdf,
        label=L"pdf $P_{%$N}$, (k=5000 samples)",
        legend=:top
    )
    plot!(plt, 0:0.01:1, x-> 1/ (π * sqrt(-(x-1)*x)),
        label=L"pdf $\frac{1}{\pi \sqrt{-(x-1)x}}$",
        linewidth=2)
    return plt
end
```

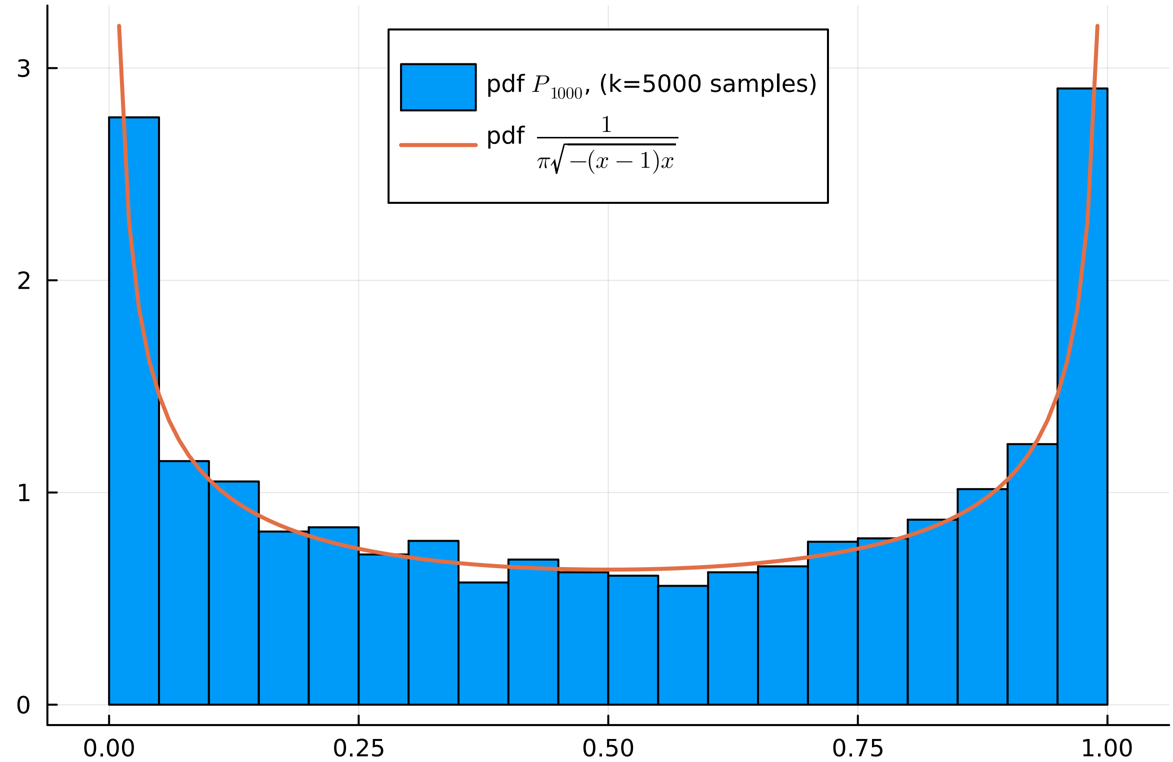
Poniżej przedstawiono unormowane wykresy aproksymowanej funkcji gęstośći prawdopodobieństwa P_N dla $N \in \{100, 1000, 10000\}$. Na wykres naniesiono też wykres gęstośći rozkłądu arcusa sinusa o dokładnym wzorze

$$\frac{d}{dx}\frac{2}{\pi}\arcsin(\sqrt{x}) = \frac{1}{\pi\sqrt{-(x-1)x}}.$$

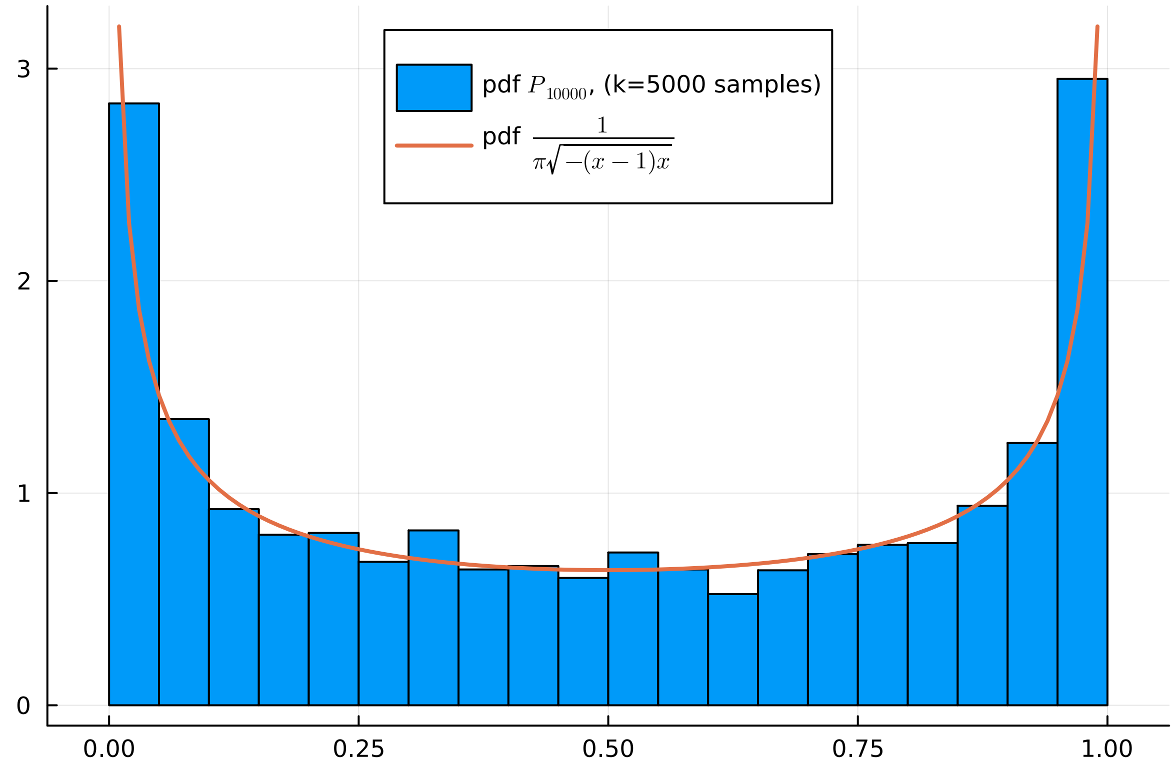
.




```
frac_over_OX(100, 5000)
```




```
• frac_over_OX(1000, 5000)
```




```
• frac_over_OX(10_000, 5000)
```

Rozkłady P_N całkiem dobrze estymują gęstość prawdopodobieństwa arcusa sinusa, a im większe N tym lepsza ta aproksymacja.

Fakt, że co drugi przedział ma większą gęstość na histogramie pdf P_{100} można wytłumaczyć skończoną liczbą wartości jakie może przyjąć $P_{100} \in \{0, 0.02, 0.04, 0.06, 0.08, 0.1, \ldots\}$ (jak pijak wejdzie nad oś OX to musi wykonać tyle samo kroków, żeby zejść pod OX, więc przebywa tam parzystą ilość czasu). Jeśli przedział $[0, 1]$ podzielimy na 20 kubełków, każdy o szerokości 0.05, to każdy z nich może przyjmować dwie ($0.\_2, 0.\_4$), lub trzy ($0.\_6, 0.\_8, 0.\_0$) z dozwolonych wartości P_{100} (mogłoby też być odwotnie, w zależności od tego gdzie wpadłyby próbki z zerową częścią setną ($0.\_0$)). Na kolejnych wykresach, tj P_{1000} oraz P_{10000} gęstość może przyjmować wielkośći, stąd bias związany z większymi gęstościami co drugiego kubełka znika.

Podobnie jak w zadaniu 2, tutaj też nasuwa się wniosek, że mając do dyspozycji jedynie generator losowych, pojedynczych bitów, można wylosować zmienną z rozkładu arcusa sinusa - wystarczy wybrać dostatecznie duże N, aby otrzymać satysfakcjonujące nas przybliżenie.

Screenshots from taken NIST tests

LCG

Bitstream input

Help

Select the input type

- Radioactive decay RNG
- ADC noise RNG
- JavaScript pseudo RNG
- (selected) Manual bitstream input

Insert Input Stream

Input bitstream

Copy to Clipboard

11101110111011011001010010110100001100110000111100000010111000111001100011110101011011
10001111110001110001000110111000001011101111011000001110000100001011101010111111110001
111001011011001000011100111000000100000001011010111000011100101001011010100100001010
101011101000000101101010011001110000011000000110100001001010011011010111001010100011
00010000100011111001100010000011101101111000101011010110111001111000110010110001111
1111111011111111011000000001111101101111110111010001010000011010001000100001100100
00100110011000001110001011001011111100101111001111100100111101110000100001011010110
0011001110010100111011101110100110010000110010100011100000000010111101000010010110
011101010011101010100110100011010001000001001011010011111010011011001110101101110001

Tests

Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.8212014203279661	Passed
2. Frequency Test within a Block	0.580884074974694	Passed
3. Runs Test	0.8056426848979531	Passed
4. Test for the Longest Run of Ones in a Block	0.8456868751391364	Passed
5. Binary Matrix Rank Test	0.6488208647235737	Passed
6. Non-overlapping Template Matching Test	0.2664319387800475	Passed
7. Overlapping Template Matching Test	0.762340445763404	Passed
8. Maurer's "Universal Statistical" Test	0.7590434662697596	Passed
9. Linear Complexity Test	0.8551234094940919	Passed
10. Serial Test	P-value 1: 0.9461955678112058 P-value 2: 0.807230829471723	Passed
11. Approximate Entropy Test	0.9701148784565543	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.9702804060204961 P-value Reverse: 0.9748225860929329	Passed
13. Random Excursions Test	0.04219578413832417	Passed

14. Random Excursions Variant Test	0.07593432372472608	Passed

MersenneTwister

Bitstream input

Help

Select the input type

- Radioactive decay RNG
- ADC noise RNG
- JavaScript pseudo RNG
- Manual bitstream input

Insert Input Stream

Input bitstream

Copy to Clipboard

10010101000001100001100001011001011110001101110101001101100100100000000000101011001
00101001110110010111011001000111100101000110001101100101100110101111011011100001001
10111101101010010110000001001011101110100110001110100101000000100110101100000000010
01000001000100001111110110010100110111011100111110110011010000010001100001001101001
10110001110111010000010110110100110010111111000001101100110000001101110111001011011
10100110100101010111000000001010000000111001101100110010001110001101000000001010111111
001001000010100000011100111110000100101111101001100110000101111101111000110010100110
1111100111001011100001101011111110111100010100110111111011011001101101000101111101010
011001000010110100011110110001011110111011100000010100001001010100010111001111000110

Tests

Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.14048159116628955	Passed
2. Frequency Test within a Block	0.7892072199878524	Passed
3. Runs Test	0.3279878771437552	Passed
4. Test for the Longest Run of Ones in a Block	0.1541131216153005	Passed
5. Binary Matrix Rank Test	0.7026539174670077	Passed
6. Non-overlapping Template Matching Test	0.6522536157854708	Passed
7. Overlapping Template Matching Test	0.35288882418532314	Passed
8. Maurer's "Universal Statistical" Test	0.5413271647888743	Passed
9. Linear Complexity Test	0.38817731391419175	Passed
10. Serial Test	P-value 1: 0.20958497778645147 P-value 2: 0.329064497695741	Passed
11. Approximate Entropy Test	0.22779360351574873	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.16196940795730996	Passed

	P-value Reverse: 0.2062009772408051	
13. Random Excursions Test	0.0013245977419050652	Failed
14. Random Excursions Variant Test	0.10886671847679363	Passed

Xoshiro

Bitstream input

Help

Select the input type

- Radioactive decay RNG
- ADC noise RNG
- JavaScript pseudo RNG
- Manual bitstream input

Insert Input Stream

Input bitstream

Copy to Clipboard

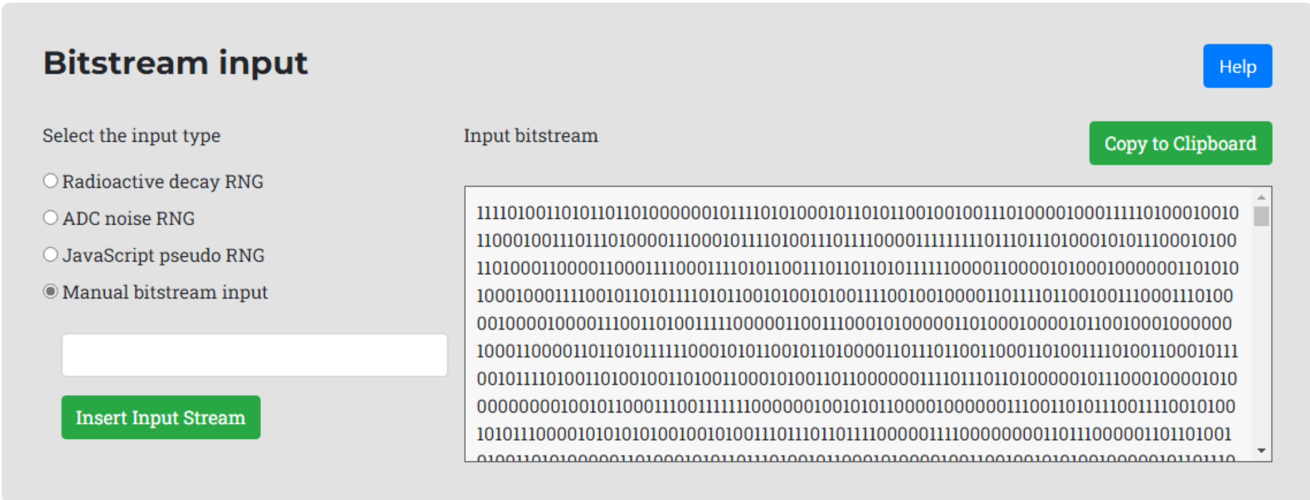
1010101110001000110010000101111011010110111011011111111000111010000010111100111101010
0011110001101011100111100011000100010101110111100110101101000110011011001000111111010
0110110111100100100101110100001110101101100110101100110000001100101101011101000111110
0101000000110010111001000101000100001001001110100010010001100101111010010010000 1
1111011011000101011000010101100111101011101011100011010000001101100100010001000110 01
0110101000010011010011100100000100110010111101110010111010101001010001100111110001 10
1001001000101111001000100001011001101110110101111010011111000001001000111011101011 01
1101100100001001011011010000000010010000010101110100011010110101111010101001110011 11
0111000011011001101101101110101010010001111001100011000010111011101111101001111101000

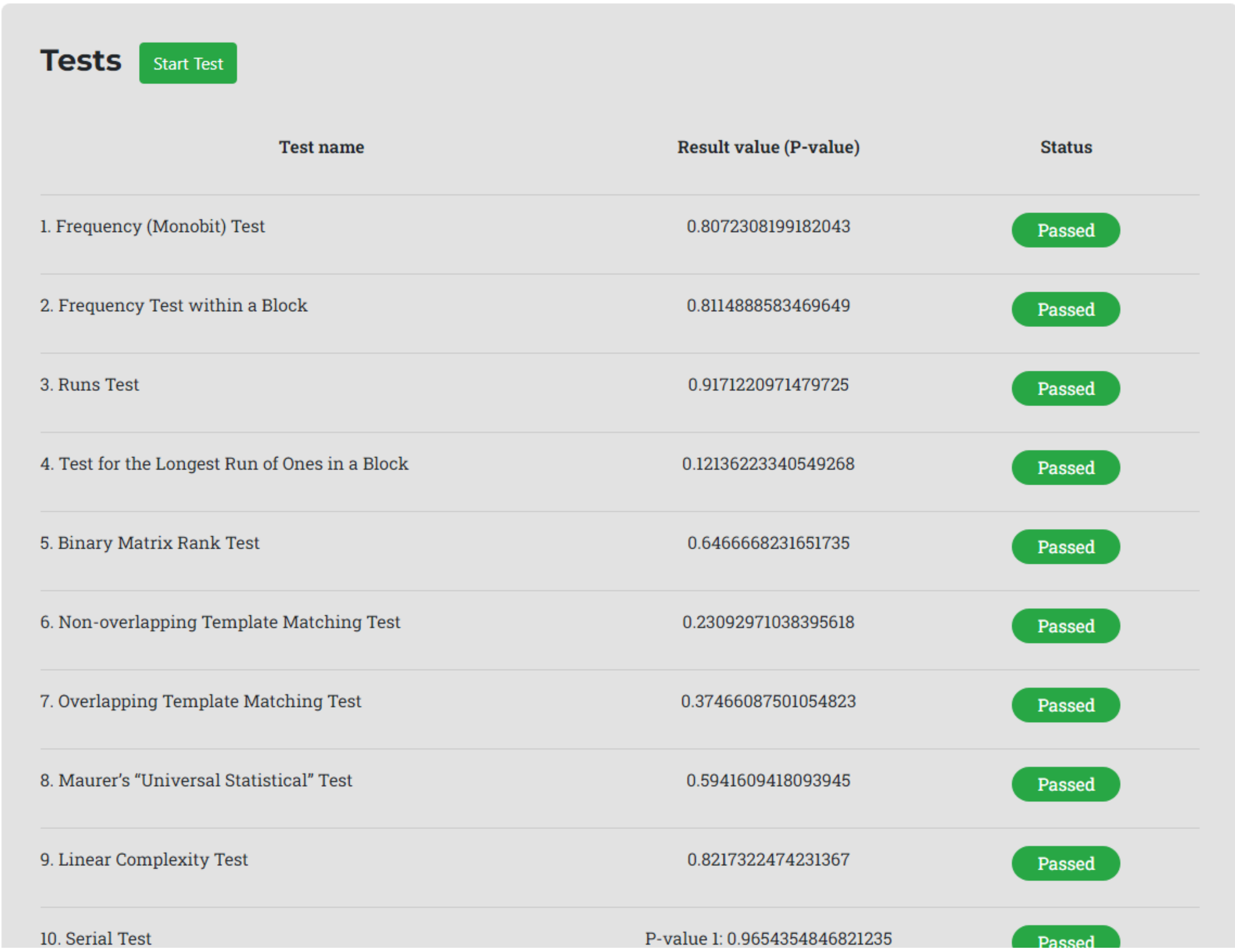
Tests

Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.8010410638013492	Passed
2. Frequency Test within a Block	0.8693294937530878	Passed
3. Runs Test	0.08799623651015187	Passed
4. Test for the Longest Run of Ones in a Block	0.5468046215971316	Passed
5. Binary Matrix Rank Test	0.18398603292879834	Passed
6. Non-overlapping Template Matching Test	0.8739971532634375	Passed
7. Overlapping Template Matching Test	0.8841404660573965	Passed
8. Maurer's "Universal Statistical" Test	0.41376816455197374	Passed
9. Linear Complexity Test	0.7972937141234042	Passed
10. Serial Test	P-value 1: 0.22682873899234965 P-value 2: 0.0883810828448054	Passed

RandomDevice





	P-value 2: 0.9171693529296843	
11. Approximate Entropy Test	0.9954073255066652	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.9698890909141935 P-value Reverse: 0.9670638355076171	Passed
13. Random Excursions Test	0.13963861323948124	Passed
14. Random Excursions Variant Test	0.12327675871463906	Passed

SHA1("Łukomski")

Bitstream input

Help

Select the input type

- Radioactive decay RNG
- ADC noise RNG
- JavaScript pseudo RNG
- Manual bitstream input

11010001011001100001110011110101111111001

Insert Input Stream

Input bitstream

Copy to Clipboard

1101000101100110000111001111010111111100100000010111010001100110011011001011001111001
11101101011000110010100000111001010110101000101110011101100000100000101111O

Tests

Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.7518296340458492	Passed
2. Frequency Test within a Block	0.5958831106435742	Passed
3. Runs Test	0.9936882896641589	Passed
4. Test for the Longest Run of Ones in a Block	0.9189277624295596	Passed
5. Binary Matrix Rank Test		Error
6. Non-overlapping Template Matching Test		Error
7. Overlapping Template Matching Test		Error
8. Maurer's "Universal Statistical" Test		Error

9. Linear Complexity Test		Error
10. Serial Test		Error
11. Approximate Entropy Test	0.8458881903464261	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.6767145733894937 P-value Reverse: 0.9708655536252708	Passed
13. Random Excursions Test		Error
14. Random Excursions Variant Test		Error