**Wojciech Typer**
279730

Kody źródłowe zostały napisane w języku Rust. Wykresy zostały wygenerowane w Pythonie poprzez bibliotekę Matplotlib. Użyto generatora liczb pseudolosowych Mersenne Twister. Każdy kod źródłowy generuje dane liczbowe, a następnie zapisuje je do pliku tekstowego, z którego następnie generowany jest wykres.

Wszystkie dane zostały uzyskane poprzez generowania d(n) i ”pobierania” pozostałych potrzebnych danych podczas tego procesu.

Interpretacja wykresów:
**1)** b(n) - z wykresu możemy odczytać, że wraz ze wzrostem ilości unr - n, liczba losowań do uzyskania pierwszej kolizji jest większa i rośnie w tempie $\sqrt{n}$ (co można odczytać z wykresu b(n) / $\sqrt{n}$ - wartości średnie na tym wykresie można aproksymować do stałej różnej od zera) . Wraz ze wzrostem ilości urn, wyniki bardziej się rozpraszają, a ich fluktuacja jest większa. Dla mniejszych ilości urn wyniki są bardziej skupione wokół średniej.

**2)** u(n) - z wykresu możemy odczytać, że średnia liczba pustych urn po wrzuceniu określonej ilości kul rośnie liniowo wraz ze wzrostem n (co możemy odczytać z wykresu u(n) / n - wartości średnie na tym wykresie można aproksymować do stałej różnej od zera). Jest to zgodne z intuicją - zwiększenie ilości urn i brak zwiększenia ilości kul zwiększa ilość pustych urn po wykonaniu eksperymentu. Wszystkie wartości są mocno skupione wokół wartości średnich.

**3)** c(n) i d(n) - z wykresów możemy odczytać, że wraz ze wzrostem liczby n (urn) liczba potrzebnych rzutów, aby osiągnąć określone warunki, rośnie w tempie n ln n. Dla c(n) (liczba rzutów potrzebnych, aby w każdej urnie była co najmniej jedna kulka) obserwujemy, że aproksymując wykres c(n)/(n ln n), wartość pozostaje stała i różna od zera. Podobną zależność widzimy dla d(n) (liczba rzutów potrzebnych, aby w każdej urnie były co najmniej dwie kule), gdzie aproksymacja d(n)/(n ln n) również pokazuje stałą wartość różną od zera.

Dodatkowo, dla większych wartości n, wyniki zarówno c(n), jak i d(n) są bardziej rozproszone wokół średniej, a fluktuacje wyników są większe. Natomiast dla mniejszych wartości n, wyniki są bardziej skupione wokół średniej, co wskazuje na mniejszą zmienność w mniejszych próbach.

**4)** d(n) - c(n) - z wykresu możemy odczytać, że wraz ze wzrostem liczby n (urn) różnica d(n) i c(n) rośnie, aczkolwiek ciężko jednoznacznie określić asymptotykę funkcji. Z wykresów (d(n) - c(n)) / (n ln n) i (d(n) - c(n)) / (n ln ln n) możemy odczytać, że aproksymując ułożenie wartości średnich otrzymujemy wartość stałą, różną od zera. Na tej podstawie możemy założyć, że asymptotyka d(n) - c(n) to n ln n lub n ln ln n. Dla mniejszych wartości n, wyniki są bardziej skupione wokół średniej, natomiast dla większych wartości wyniki są bardziej rozproszone wokół średniej.

**Birthday paradox** - paradoks związany z rachunkiem prawdopodobieństwa. Odpowiada na pytanie, ile potrzeba zgromadzić osób, aby prawdopodobieństwo znalezienia dwóch z taką samą datą urodzin (dzień i miesiąc) wynosiła ½? Większość osób na to pytanie odpowiada, że ilość osób powinna wynosić 366 / 2 = 183. Poprawną odpowiedzią na to pytanie jest 23 - zaskakująco mało liczba, stąd paradoks w nazwie. W naszym zadaniu wyznaczenie b(n) jest równoważne birthday paradox, na wykresie przedstawiającym tą funkcję możemy zauważyć, że moment pierwszej kolizji następował -zazwyczaj- znacznie szybciej niż wynosi połowa n.

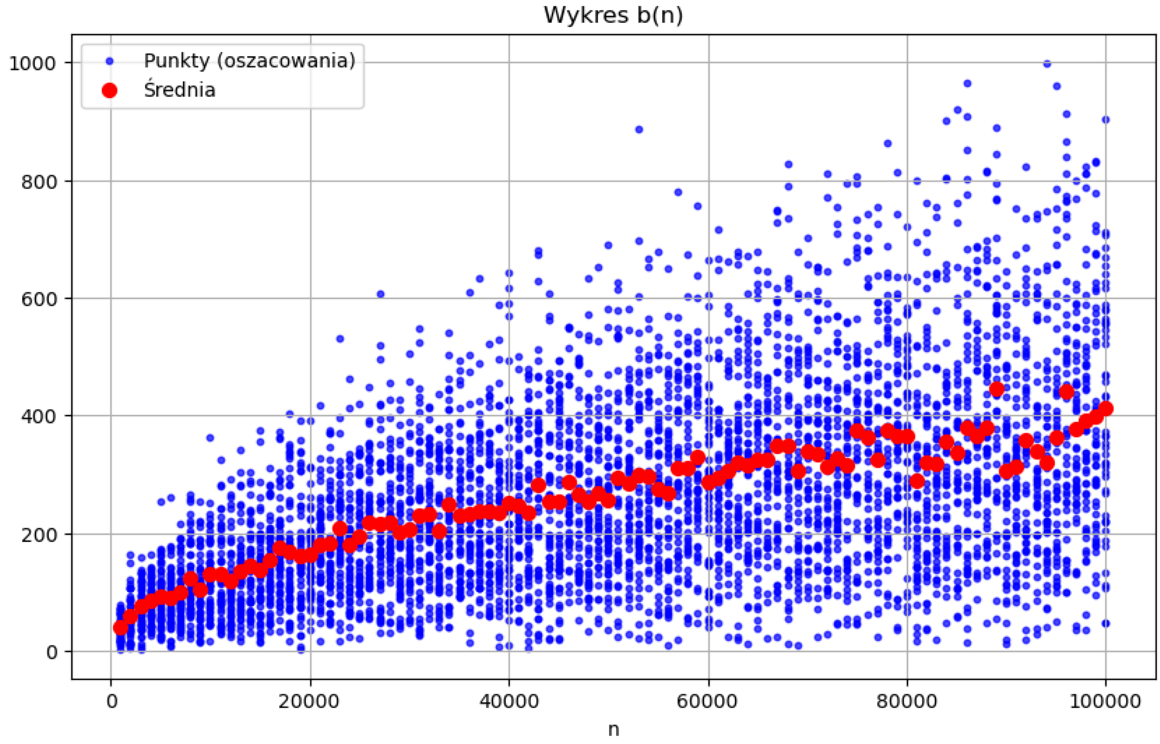
**Cupon collector's problem** - problem probabilistyczny polegający na przewidzeniu jak długo należy zbierać kupony, aby zebrać je wszystkie. Zebranie jednego kuponu polega na wylosowaniu go. W naszym zadaniu wyznaczenie c(n) jest równoważne problemowi kolekcjonera kuponów - zliczamy po ilu rzutach w każdej urnie będzie znajdować się co najmniej jedna kulka.

**Jakie znaczenie ma birthday paradox w kontekście funkcji hashujących i kryptograficznych funkcji hashujących?**
Funkcje hashujące mapują dużą ilość danych wejściowych do skończonego zakresu wartości. Ze względu na ograniczony zakres możliwych hash, zgodnie z zasadą szufladkową Dirichleta, kolizje są nieuniknione. Birthday paradox pozwala oszacować po jakim czasie dojdzie do pierwszych kolizji przy losowych próbach.

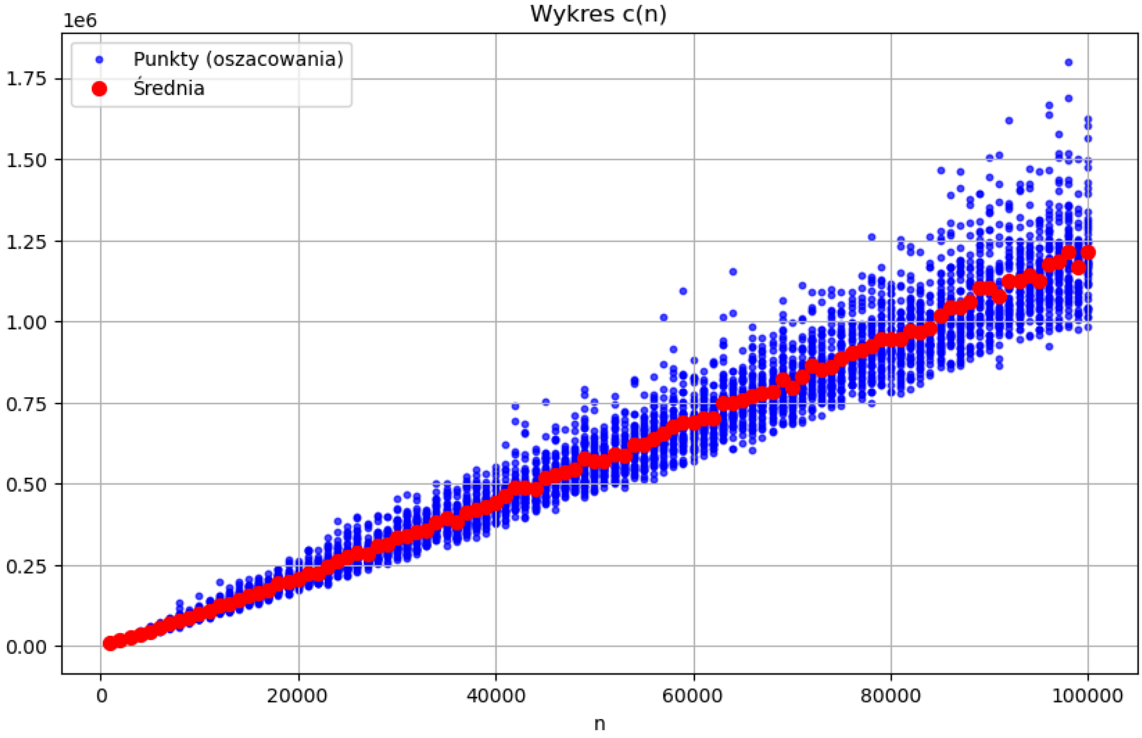
Kryptograficzne funkcję hashujące mają kluczowe znaczenie w kontekście bezpieczeństwa systemów informatycznych. Paradoks urodzinowy wpływa na:

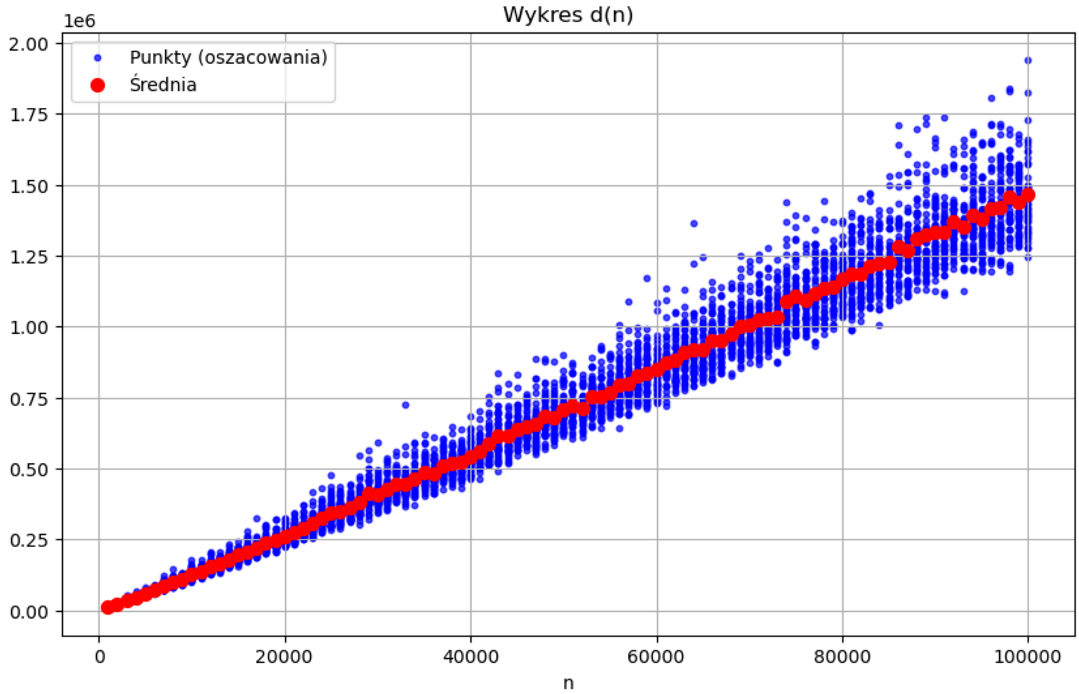
**Atak urodzinowy:** jego celem jest znalezienie kolizji funkcji hashujących. U jego podstaw leży birthday paradox, dzięki któremu można oczekiwać, że kolizja zostanie znaleziona znacznie szybciej niż sugerowałby to rozmiar przeciwdziedziny funkcji hashującej.

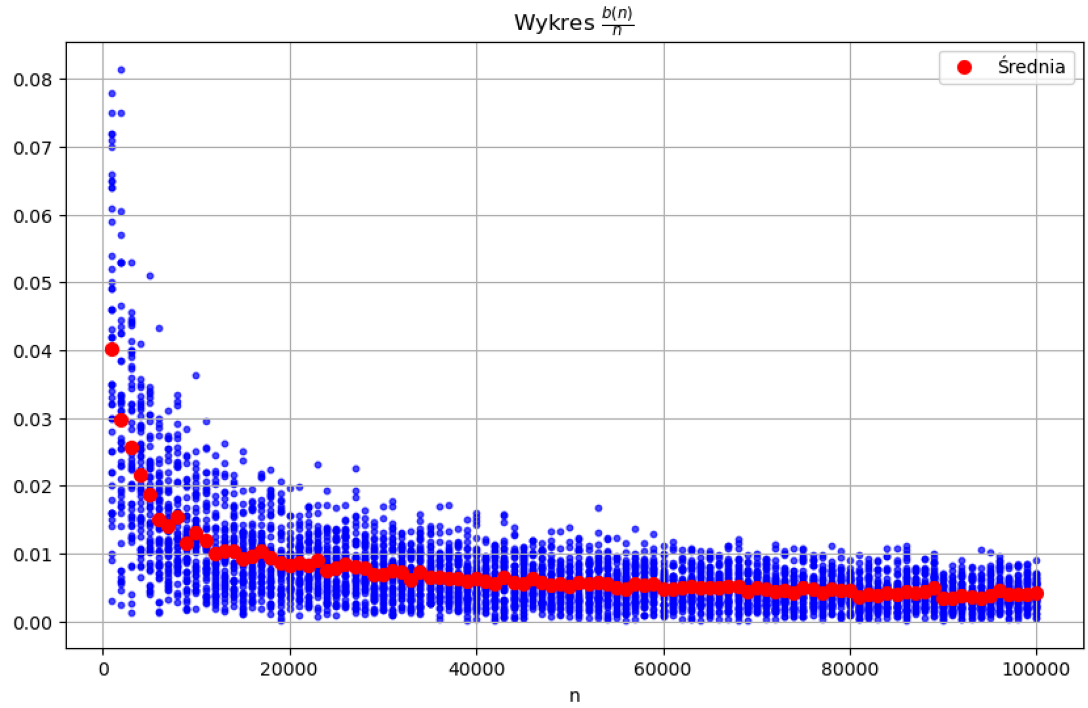
**Projektowanie funkcji hashującej:** funkcja hashująca powinna być (w miarę możliwości) odporna na atak urodzinowy. Algorytm powinien być zaprojektoany w taki sposób, aby skutecznie generować losowy rozkład wyników hash, minimalizując ryzyko kolizji.


Wykres b(n)
Punkty (oszacowania)
Średnia
1000
800
600
400
200
0
0
20000
40000
60000
80000
100000
n

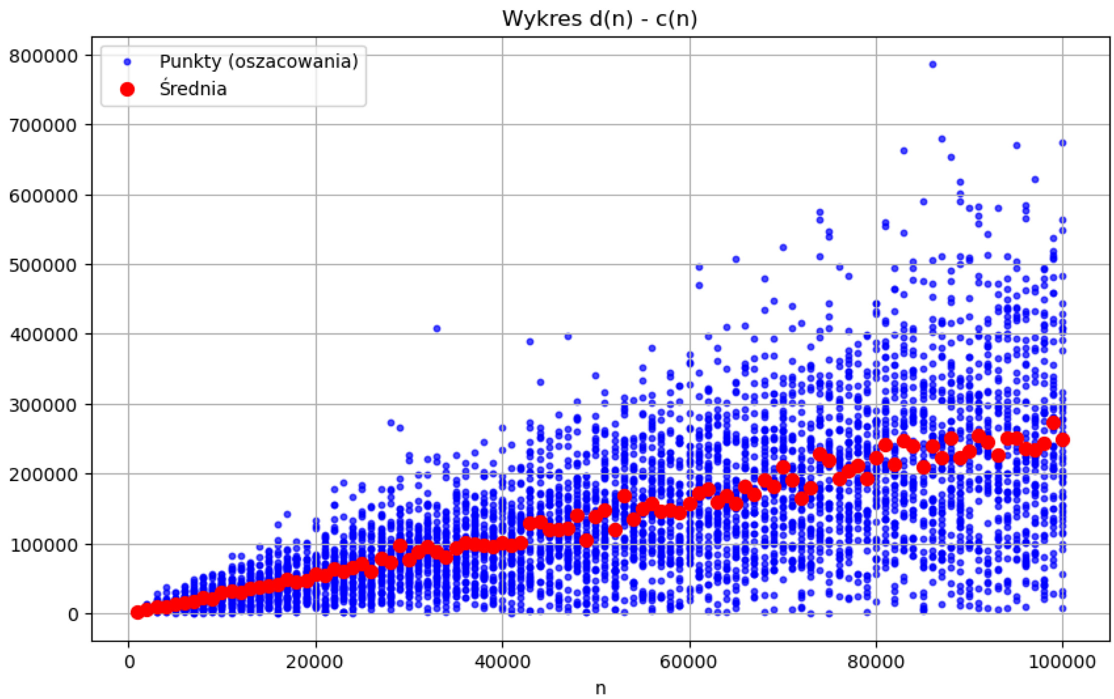

Wykres u(n)
Punkty (oszacowania)
Średnia
35000
30000
25000
20000
15000
10000
5000
0
0
20000
40000
60000
80000
100000
n


Wykres c(n)
1e6
Punkty (oszacowania)
Średnia
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20000
40000
60000
80000
100000
n

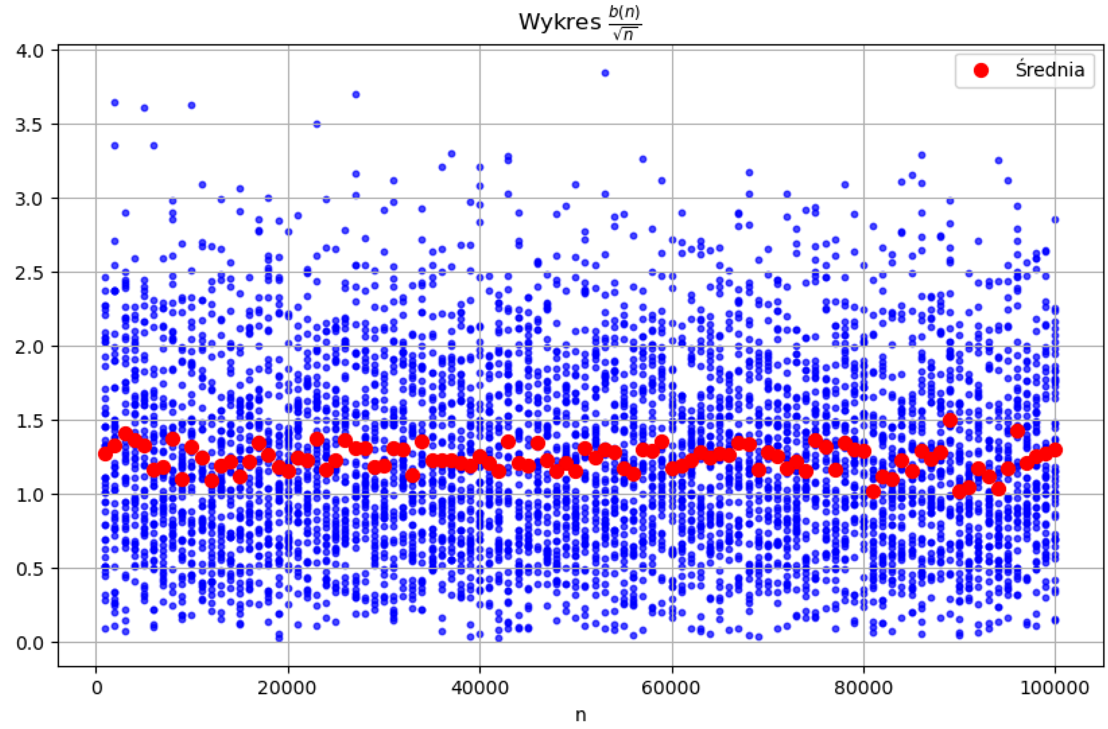

Wykres d(n)
1e6
Punkty (oszacowania)
Średnia
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20000
40000
60000
80000
100000
n

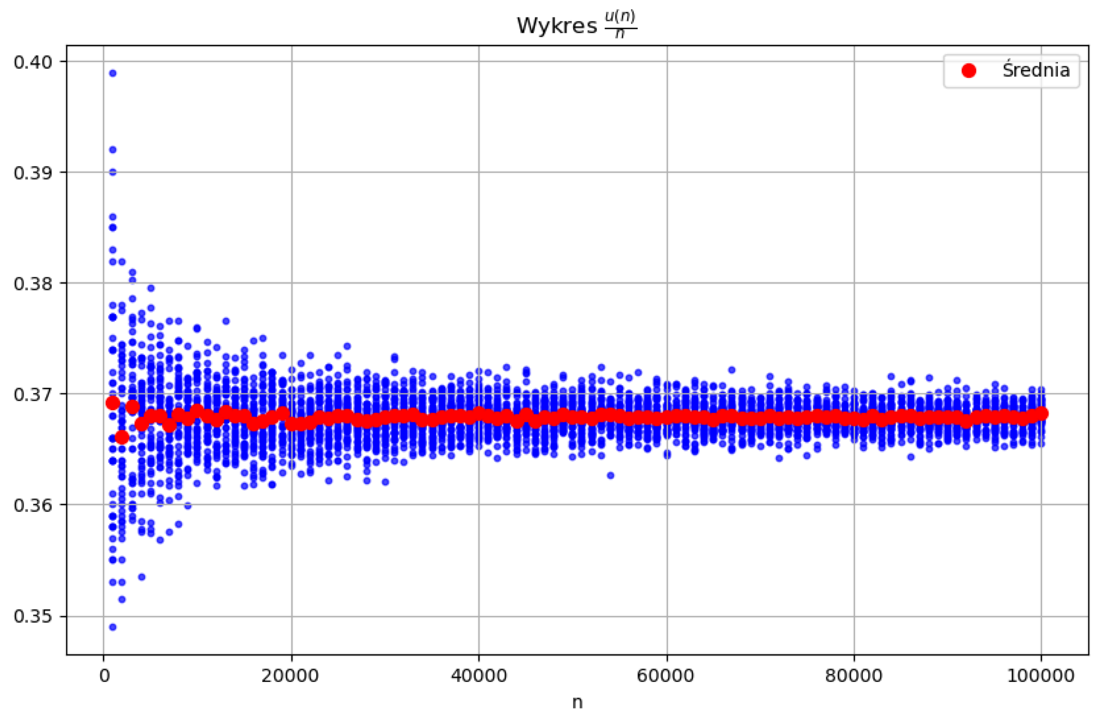


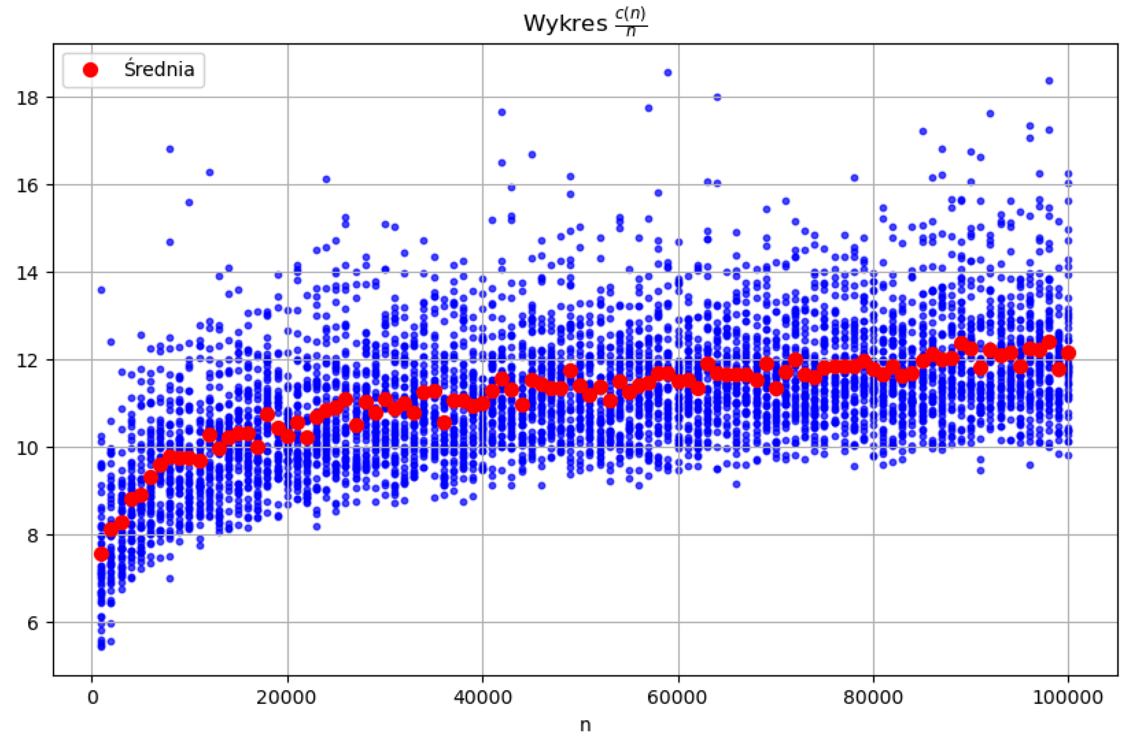
Wykres $\frac{b(n)}{n}$
Średnia
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20000
40000
60000
80000
100000
n

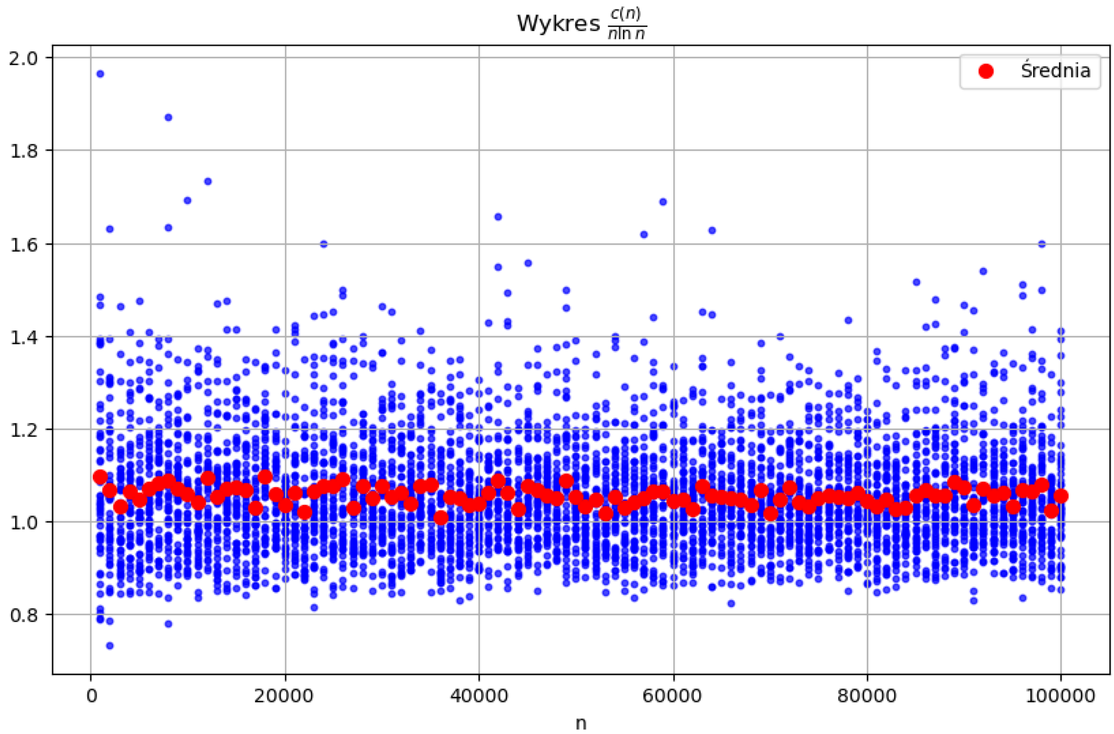
Wykres d(n) - c(n)
Punkty (oszacowania)
Średnia
800000
700000
600000
500000
400000
300000
200000
100000
0
0
20000
40000
60000
80000
100000
n


Wykres $\frac{b(n)}{\sqrt{n}}$
Średnia
n

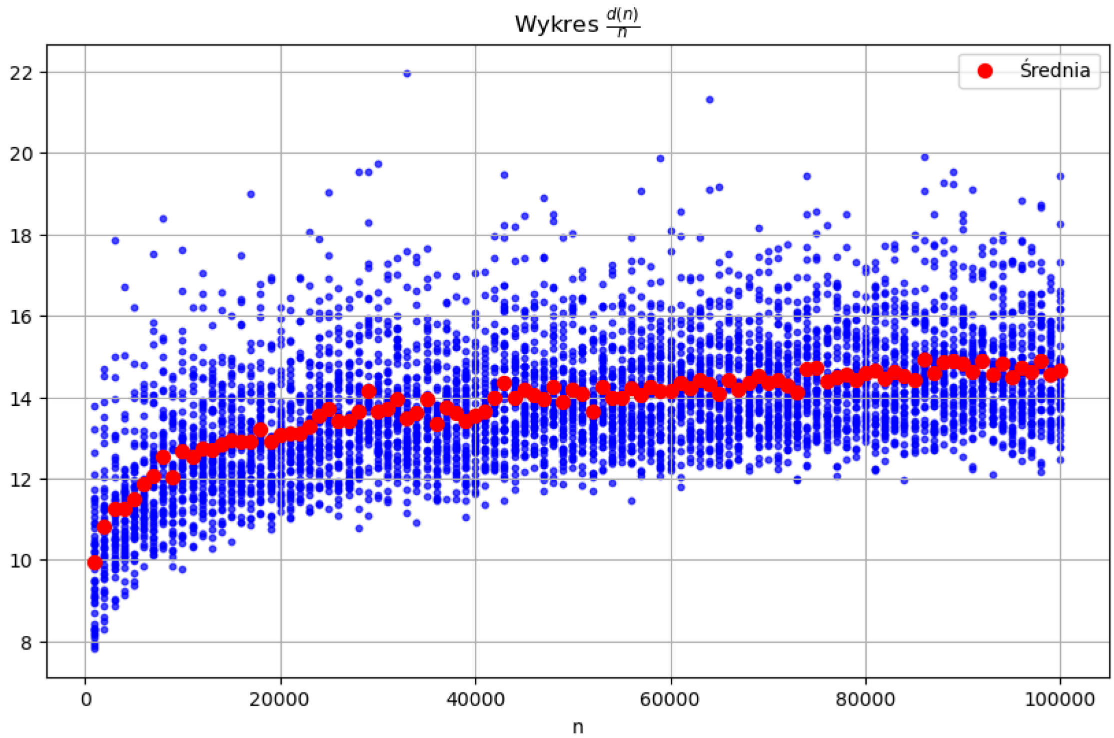

Wykres $\frac{u(n)}{n}$
Średnia
n



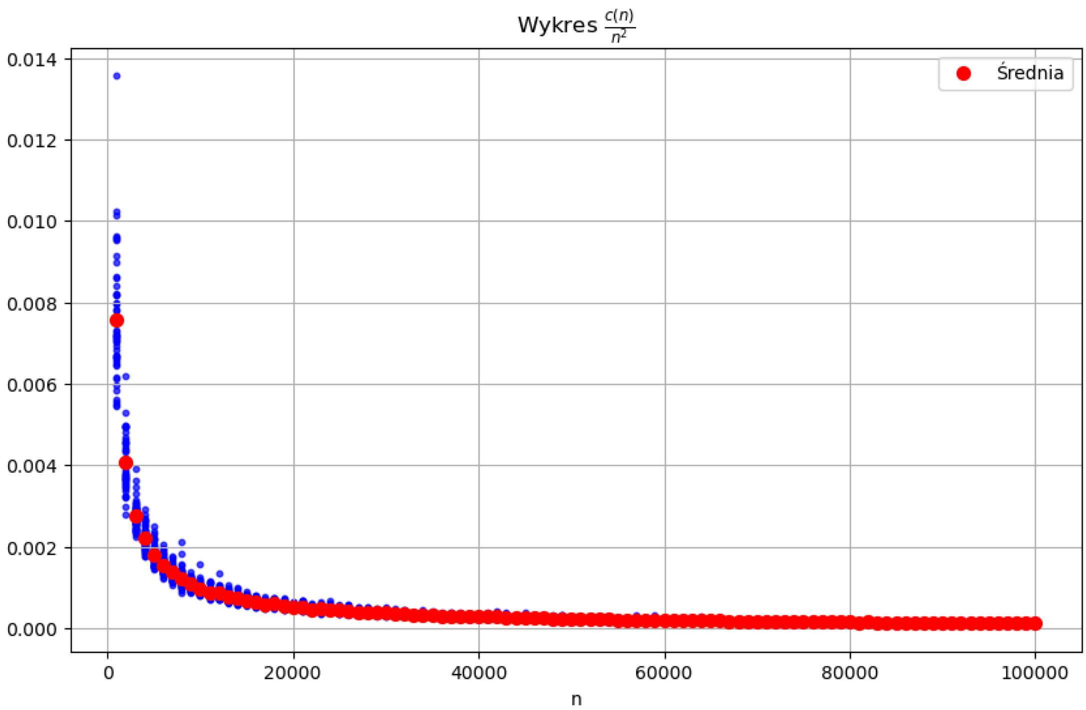
Wykres $\frac{c(n)}{n}$
Średnia
18
16
14
12
10
8
6
0
20000
40000
60000
80000
100000
n

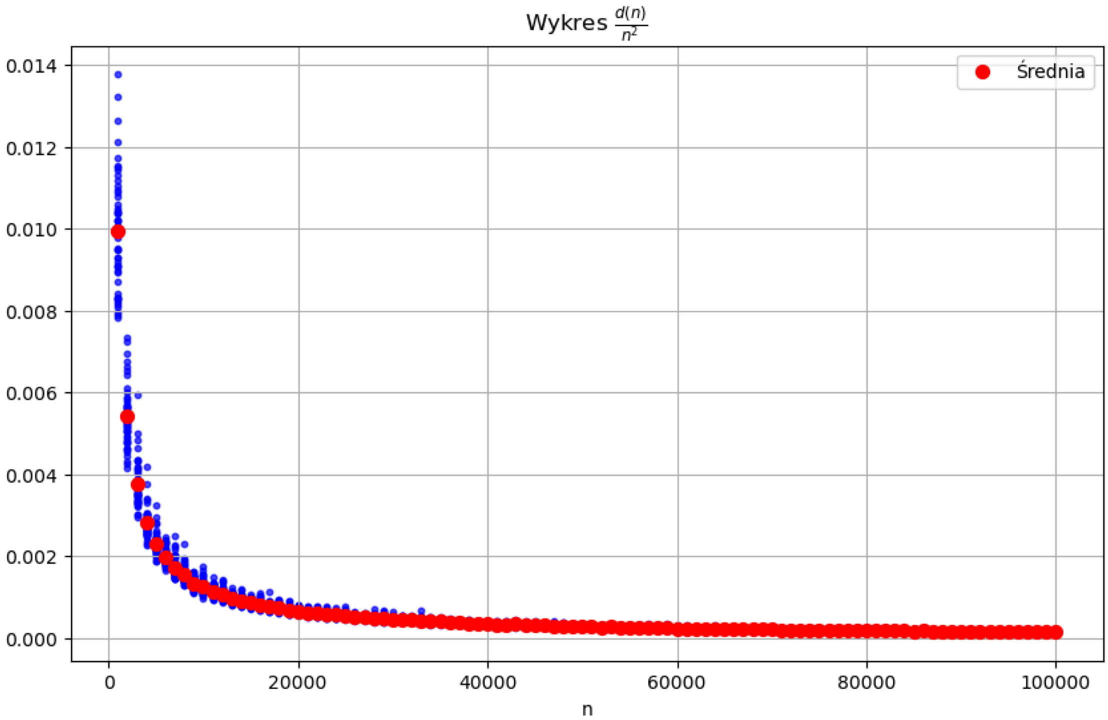
Wykres $\frac{c(n)}{n \ln n}$
Średnia
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
20000
40000
60000
80000
100000
n



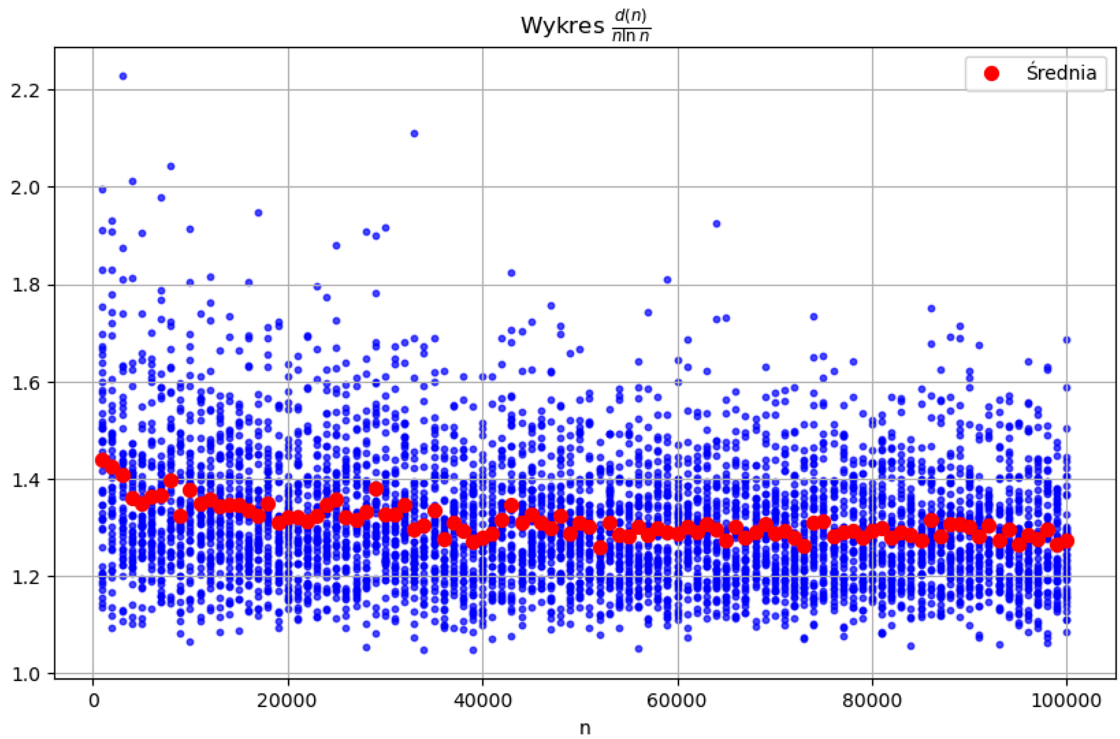
Wykres $\frac{d(n)}{n}$
Średnia
22
20
18
16
14
12
10
8
0
20000
40000
60000
80000
100000
n

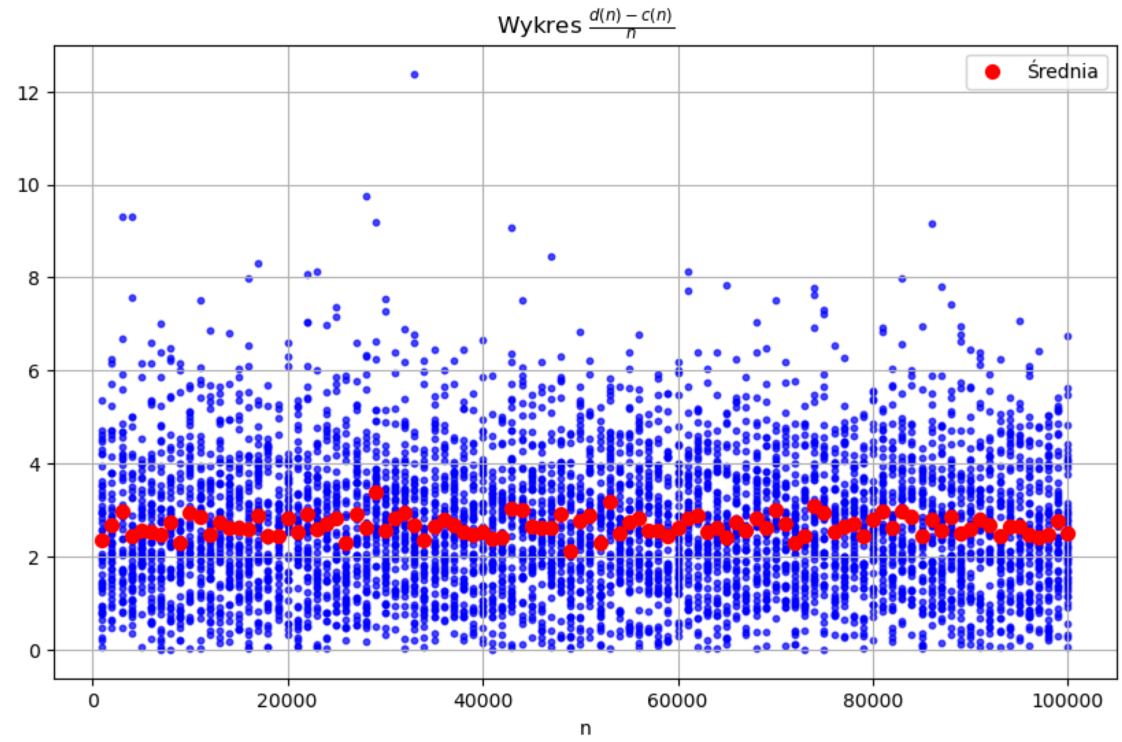
Wykres $\frac{c(n)}{n^2}$
Średnia
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
20000
40000
60000
80000
100000
n



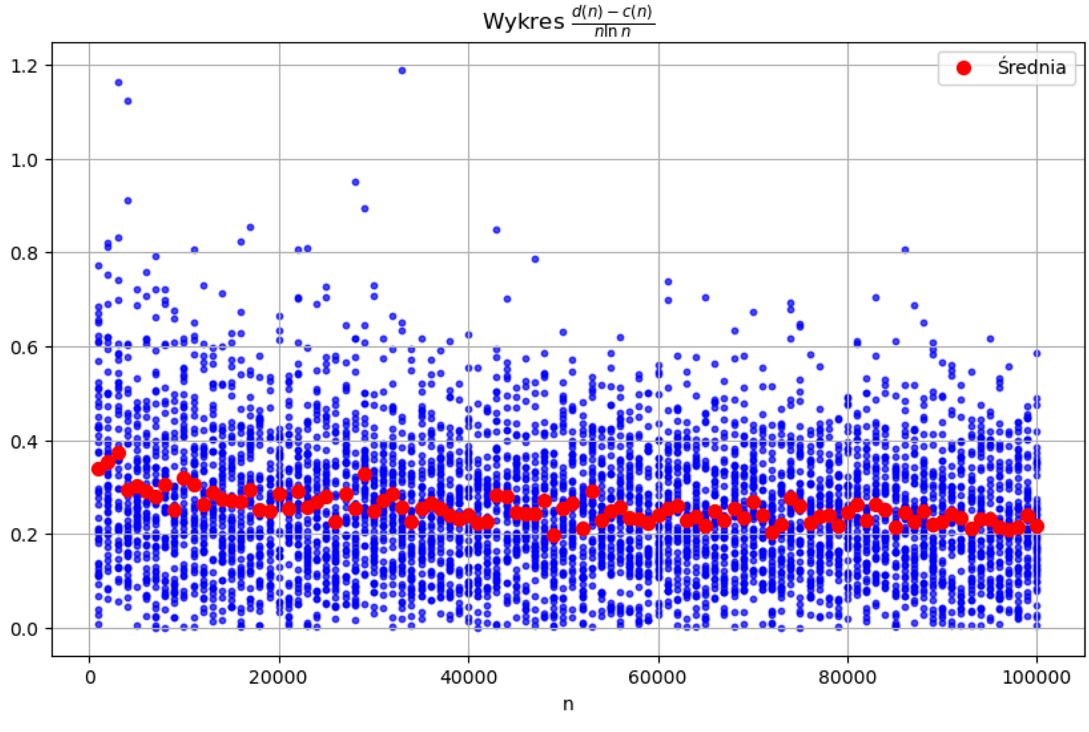
Wykres $\frac{d(n)}{n^2}$
Średnia
n

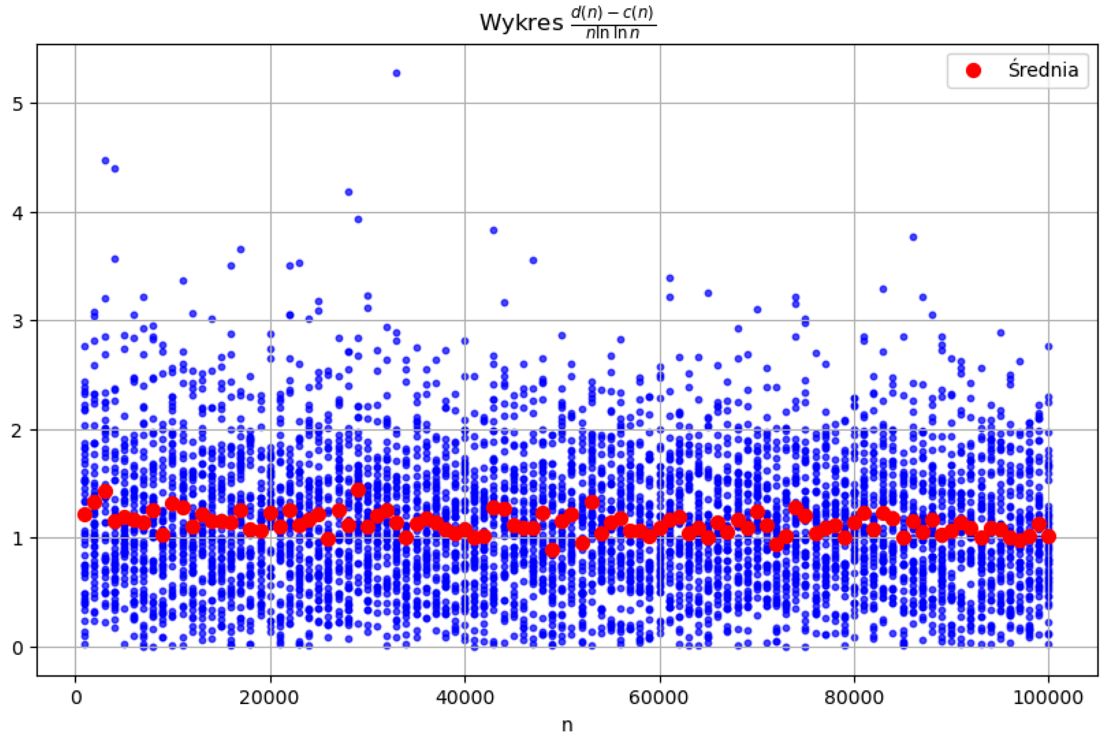
Wykres $\frac{d(n)}{n \ln n}$
Średnia
n



Wykres $\frac{d(n)-c(n)}{n}$
Średnia
n

Wykres $\frac{d(n)-c(n)}{n \ln n}$
Średnia
n


Wykres $\frac{d(n)-c(n)}{n \ln \ln n}$
Średnia
n
0
20000
40000
60000
80000
100000
0
1
2
3
4
5