Metody Probabilistyczne i Statystyka – zadanie domowe nr 2

Szymon Hładyszewski

Kody źródłowe zostały napisane w języku Java w oparciu o Maven, aby można było zastosować generator Mersenne Twister. Wszelkie otrzymane wyniki są zapisywane w pliku .csv, a wykresy zostały wygenerowane w Excelu.

Opis i rezultaty poszczególnych wykresów, koncentracja uzyskanych wyników i wnioski:

Na pierwszych pięciu poniżej przedstawionych wykresach małe niebieskie kropki oznaczają pojedyncze wyniki dla danej ilości urn. Większe czerwone kropki symbolizują średni wynik z pięćdziesięciu powtórzeń eksperymentu dla każdej poszczególnej ilości koszyków. Na pozostałych wykresach kropki dotyczą jedynie średnich wartości dla każdej iteracji ilości tychże urn.

1. Bn – na wykresie można dostrzec, że wraz ze wzrostem liczby koszyków – n, liczba rzutów kul do momentu uzyskania kolizji również rośnie. Jednakże wzrost ten jest zadziwiająco coraz mniejszy. Oprócz tego widoczny jest niemały rozrzut, zwłaszcza dla dużych n.
2. Un – wykres ten rośnie wręcz liniowo, a pojedyncze próby są do siebie bardzo podobne, przez co każda z nich jest bardzo zbliżona do wartości średniej dla każdego n.
3. Cn oraz Dn – wykresy zgodne z przewidywaniami, widoczne dla obu z nich rezultaty wskazują, że rozrzut dla poszczególnych eksperymentów jest stosunkowo niewielki.
4. Dn – Cn – podobnie jak przy Bn, widać tutaj duży rozrzut, zwłaszcza przy wynikach dla dużej ilości urn.

Asymptotyka:

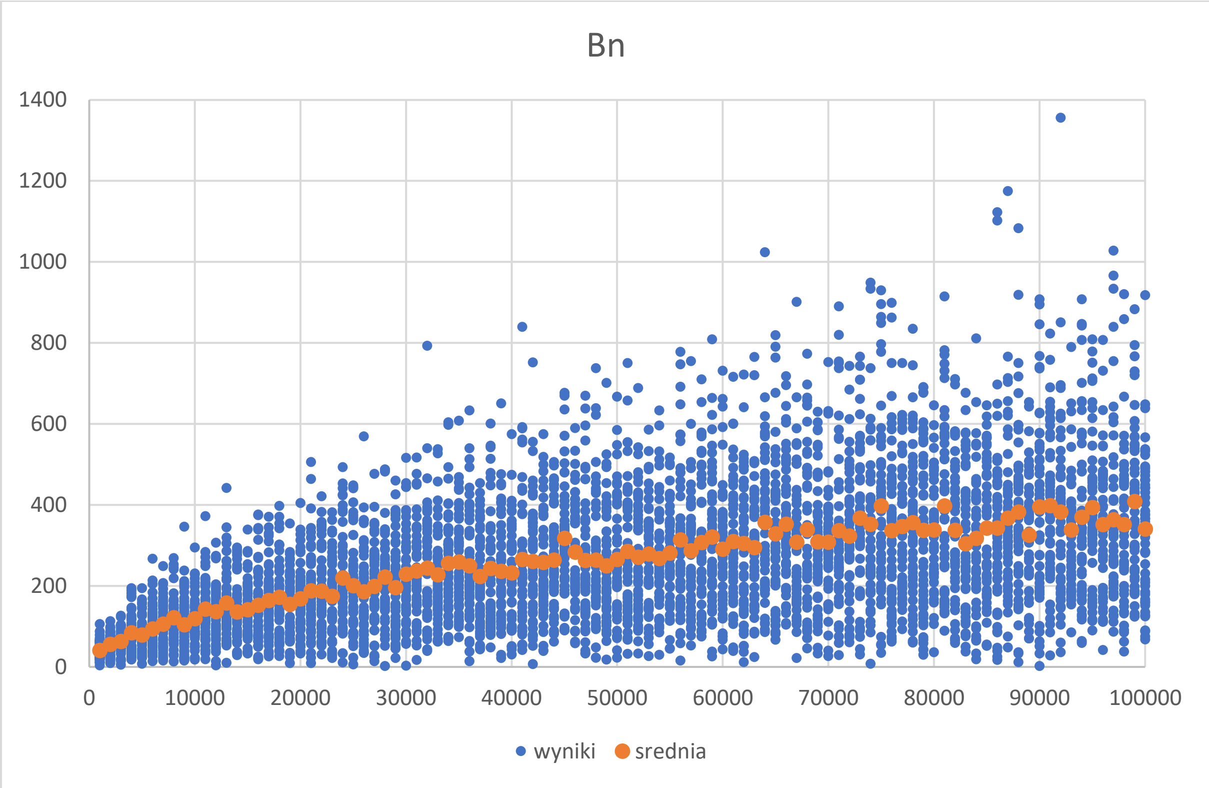
1. b(n) – rośnie w tempie sqrt(n), co wynika z wykresu b(n)/sqrt(n), gdzie wartości średnie można przybliżyć do stałej różnej od zera oraz klasycznego rozwiązania problemu paradoksu urodzinowego.
2. u(n) – wzrost wynosi n, co również wynika z wykresu funkcji u(n)/n (wyniki można aproksymować do stałej różnej od zera).
3. c(n) – aproksymacja wykresu c(n)/(n*ln(n)) posiada wartości stałe i różne od zera, zatem c(n) rośnie w tempie n*ln(n), co jest zgodne z rozwiązaniem problemu kolekcjonera kuponów.
4. d(n) – identyczna sytuacja jak w przypadku c(n).
5. d(n)-c(n) – rośnie w tempie n*ln(ln(n)) na podstawie wykresu (d(n)-c(n))/(n*ln(ln(n))).

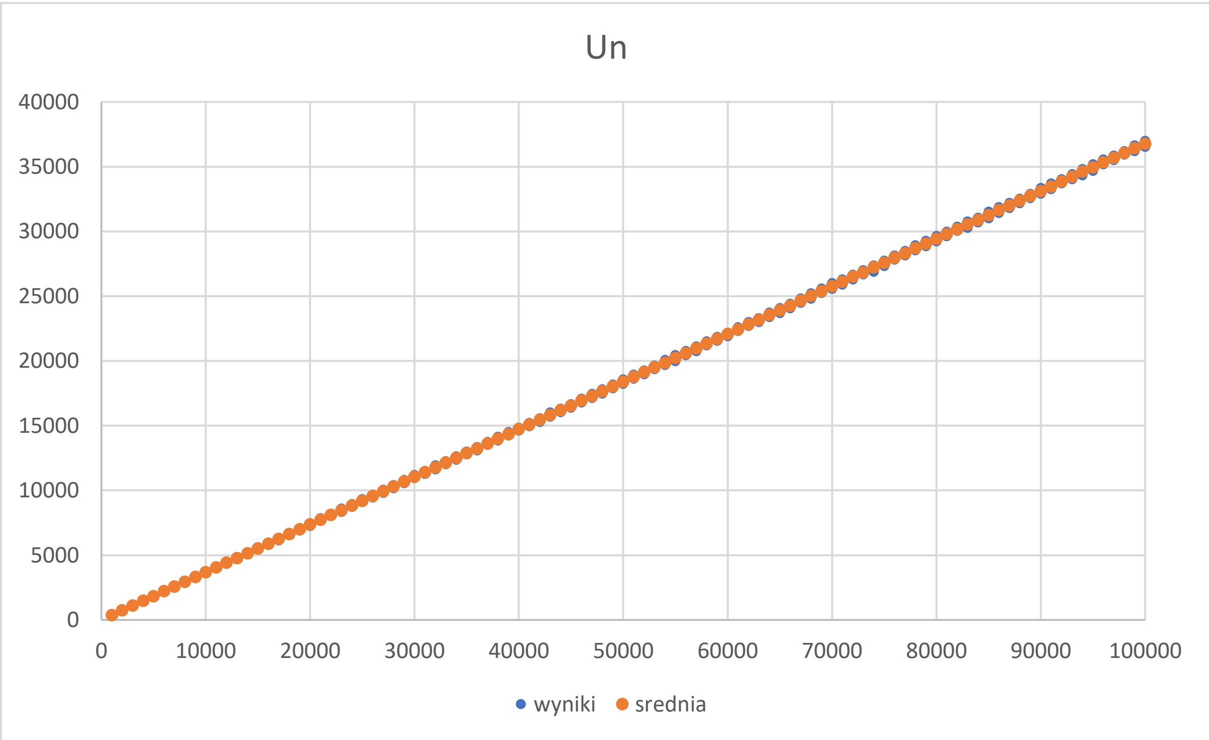
Birthday paradox – nazwa odnosi się do paradoksu, w którym szacujemy, ile potrzeba osób w grupie, aby była spora szansa na to, że znajdują się w niej dwie osoby o takiej samej dacie urodzin. Okazuje się, że liczba ta jest znacznie mniejsza niż się to wydaje. Analogicznie jest z wystąpieniem kolizji podczas wrzucania kul do urn – z każdym kolejnym powiększaniem ilości koszyków o tysiąc, moment uzyskania jej opóźnia się dosłownie o kilka rzutów.

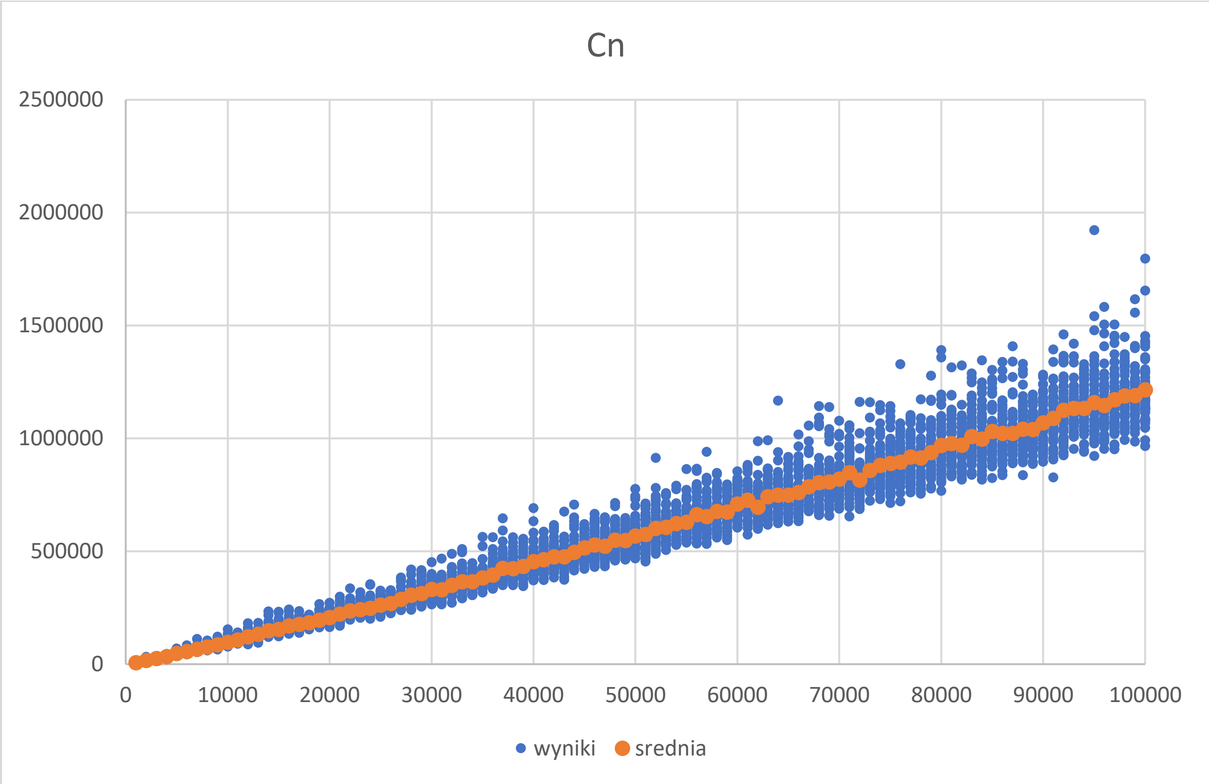
Coupon collector’s problem – użycie tej nazwy wynika z problemu jak długo kolekcjoner musi zbierać kupony, aby uzyskać je wszystkie, gdzie każdy z nich jest wydawany losowo. Analogiczna sytuacja ma miejsce przy szacowaniu ilości rzutów potrzebnej do zapełnienia każdej urny przynajmniej jedną kulą.

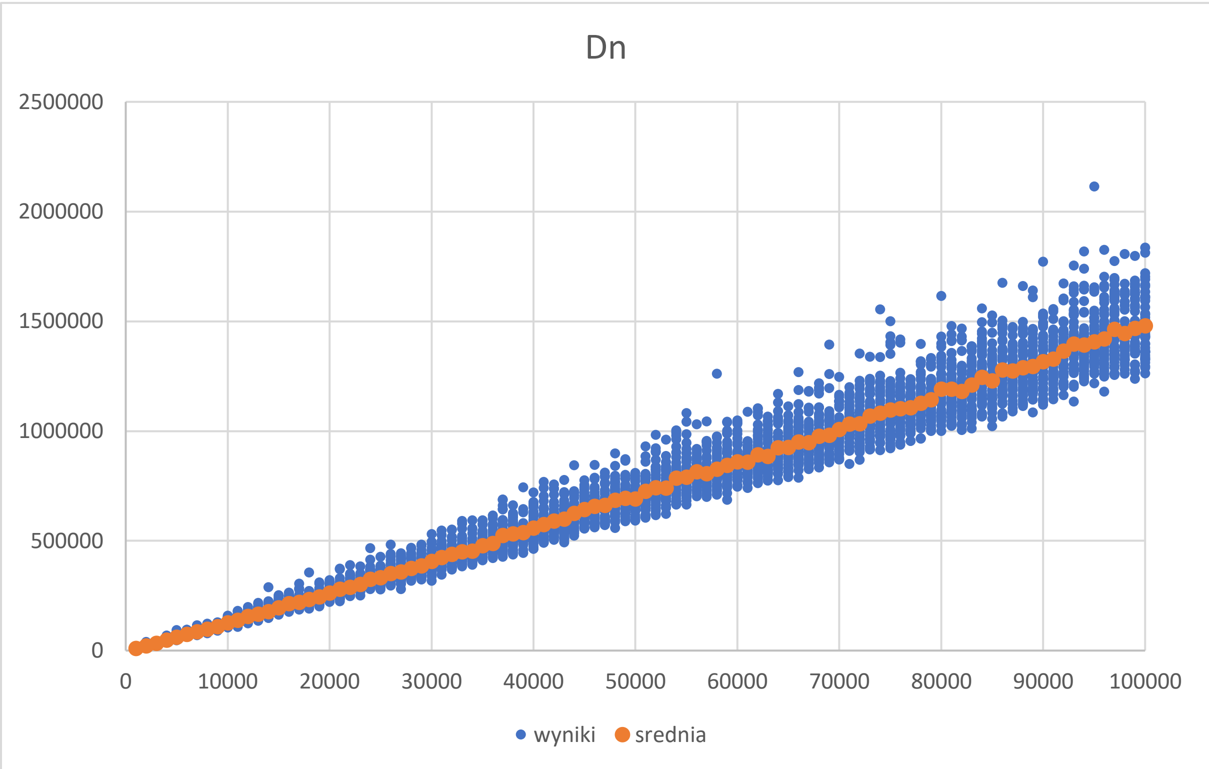
Znaczenie birthday paradox przy funkcjach hashujących i kryptograficznych funkcjach hashujących:

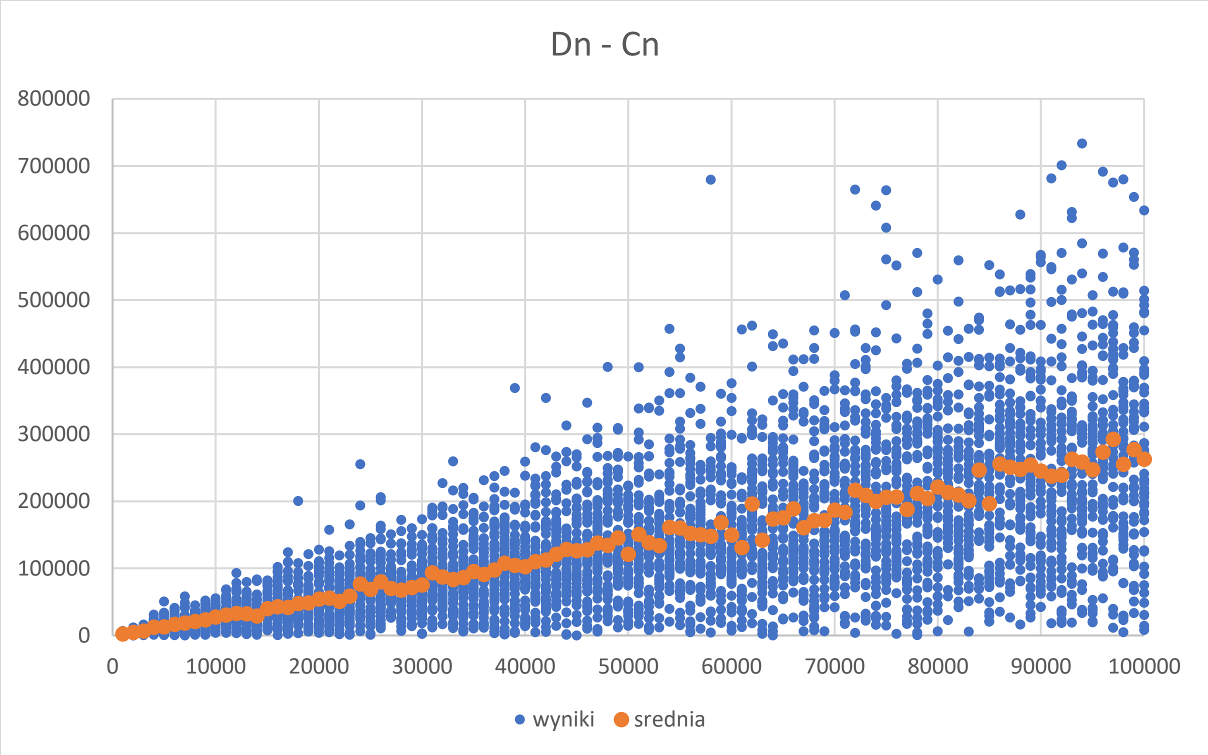
Paradoks ten jest tutaj dosyć istotny, ponieważ występuje tutaj analogia pomiędzy znalezieniem drugiej osoby z tą samą datą urodzin (birthday paradox) oraz znalezieniem danych generujących identyczny hash, który chcemy wydobyć (funkcja hashująca).

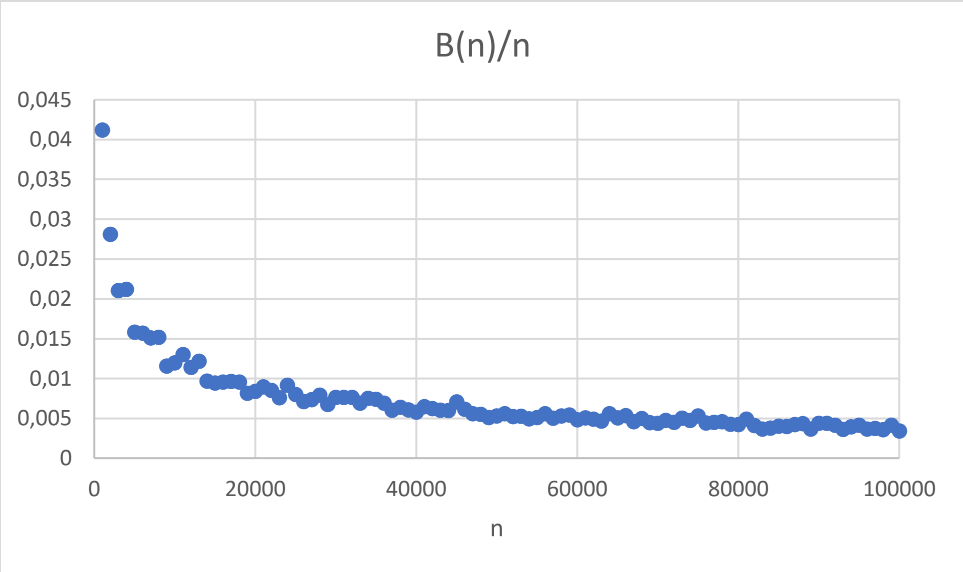

Bn
1400
1200
1000
800
600
400
200
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
wyniki
srednia

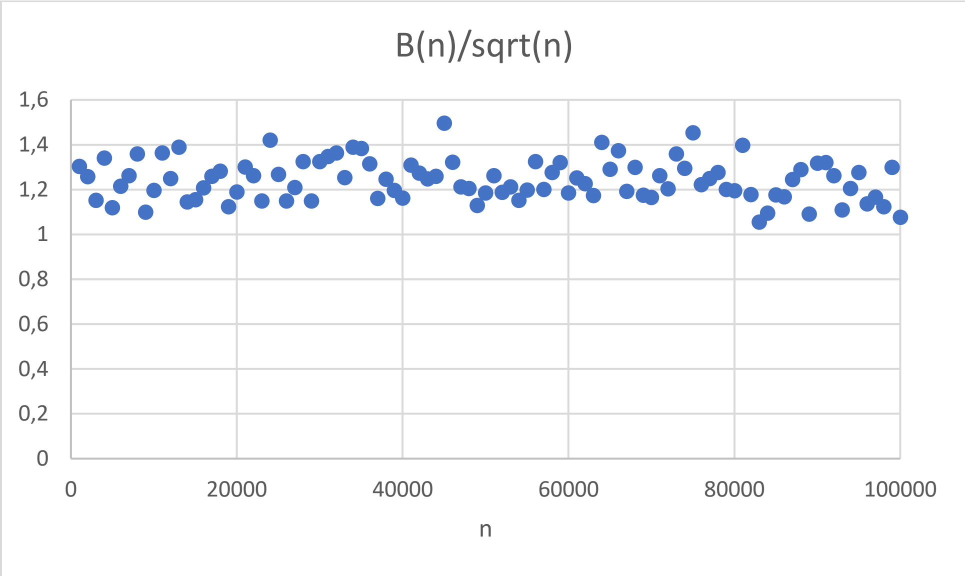

Un
40000
35000
30000
25000
20000
15000
10000
5000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
wyniki
srednia

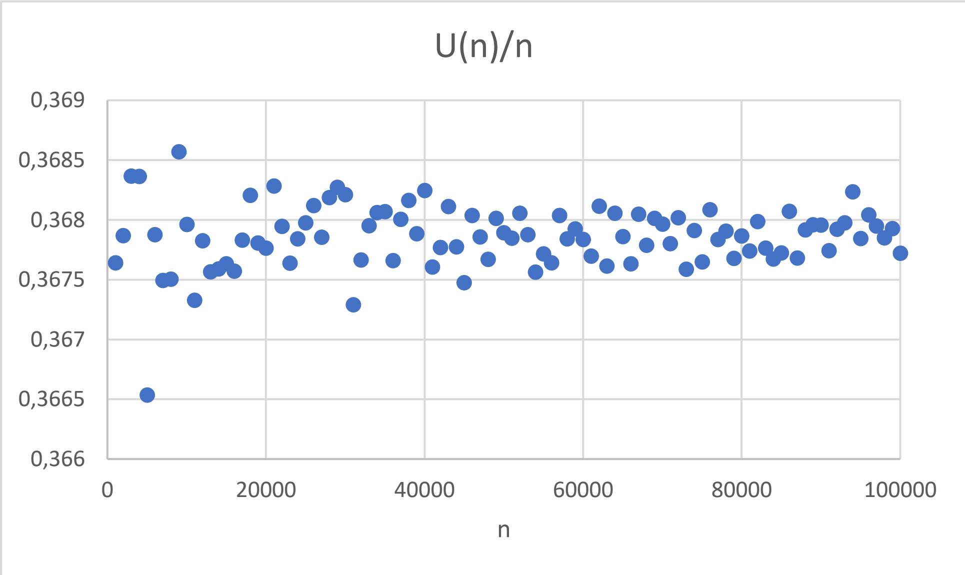

Cn
2500000
2000000
1500000
1000000
500000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
wyniki
srednia

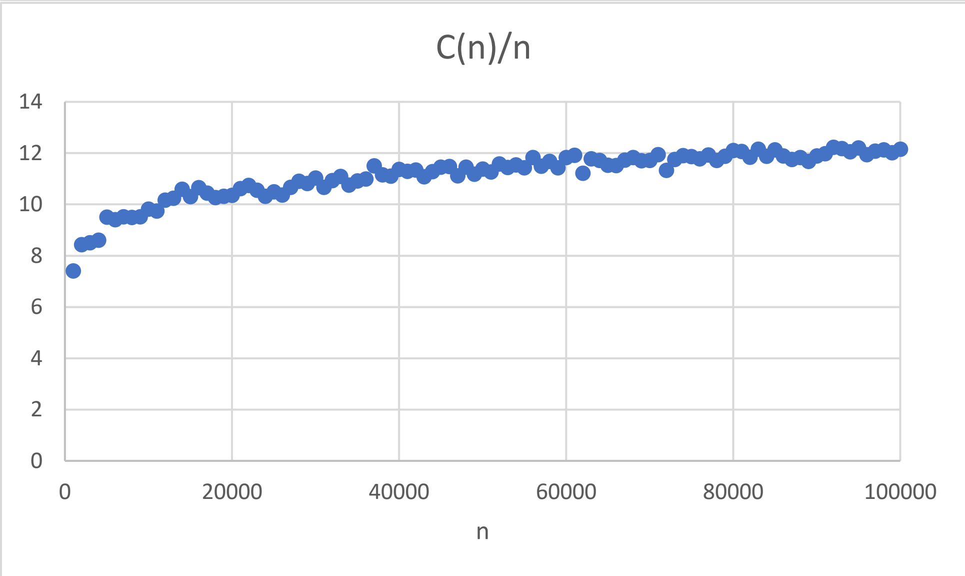

Dn
2500000
2000000
1500000
1000000
500000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
wyniki
srednia

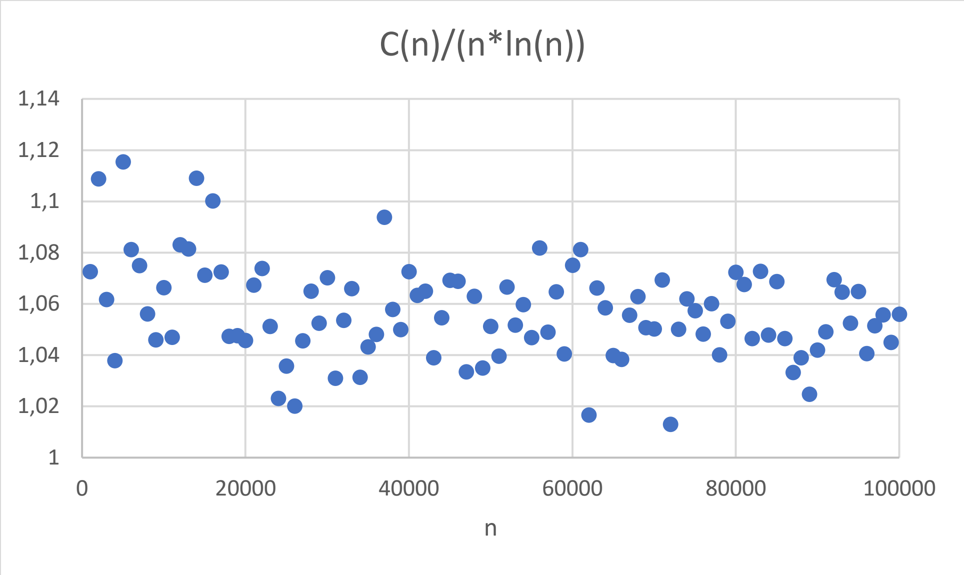

Dn - Cn
800000
700000
600000
500000
400000
300000
200000
100000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
wyniki
srednia

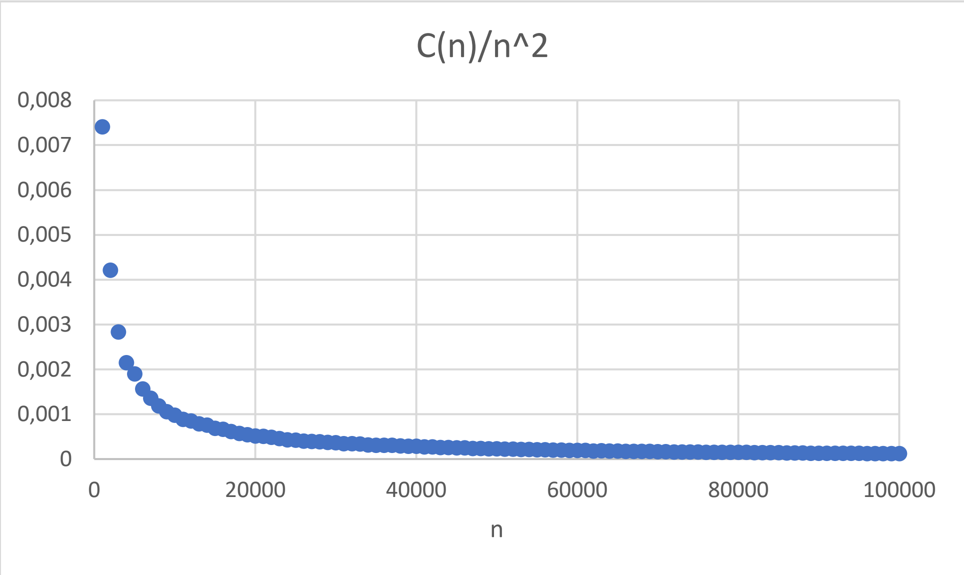

B(n)/n
0,045
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
0
20000
40000
60000
80000
100000
n

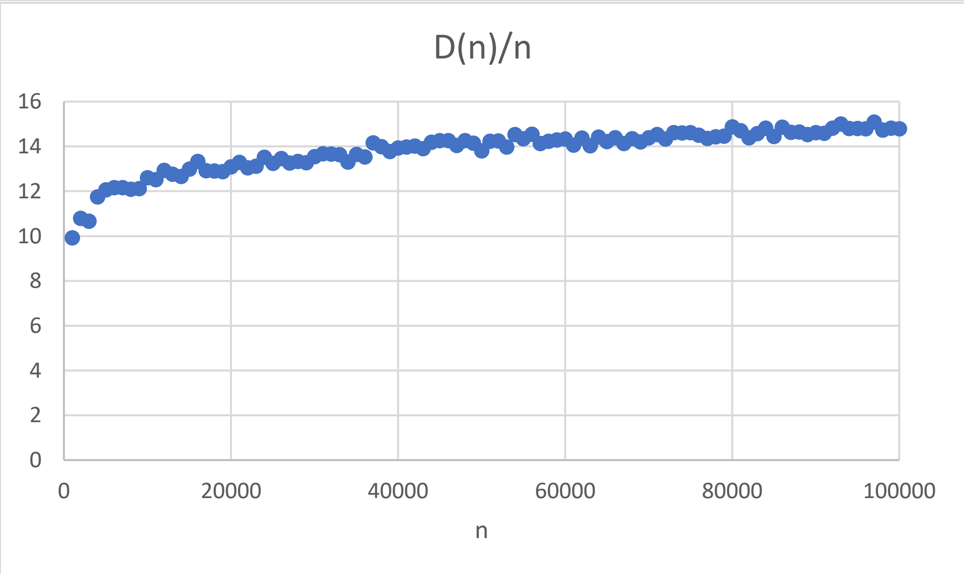

B(n)/sqrt(n)
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
0
20000
40000
60000
80000
100000
n

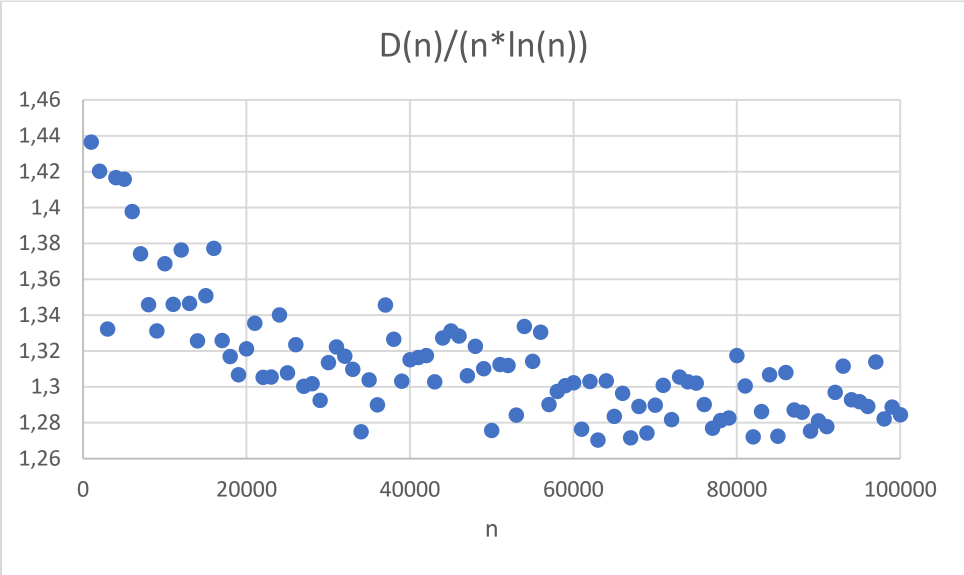

U(n)/n
0,369
0,3685
0,368
0,3675
0,367
0,3665
0,366
0
20000
40000
60000
80000
100000
n

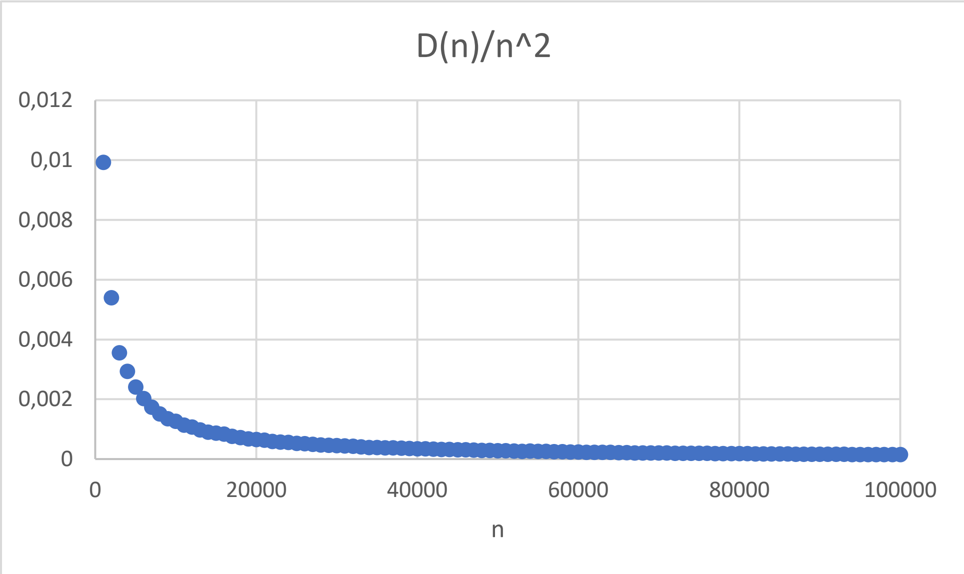

C(n)/n
14
12
10
8
6
4
2
0
0
20000
40000
60000
80000
100000
n

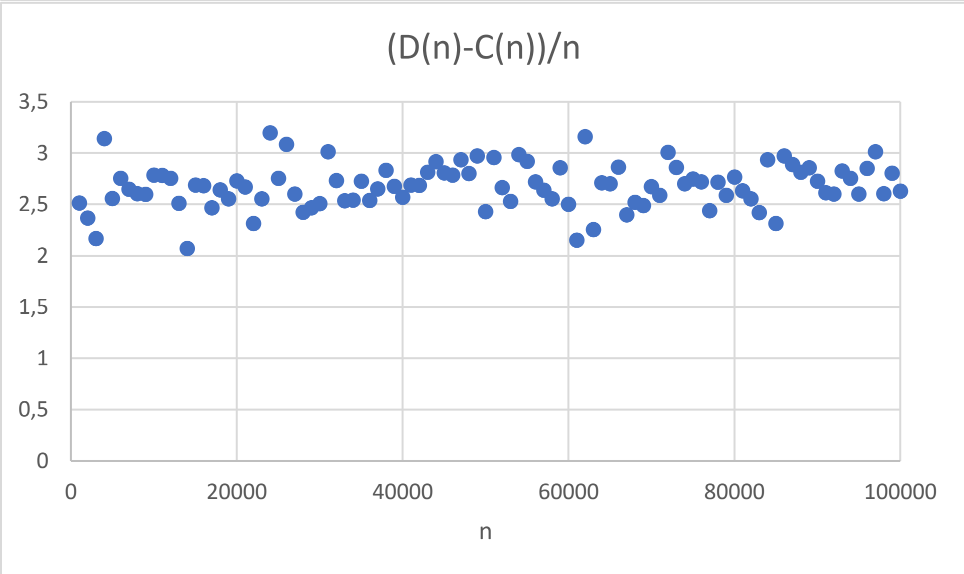

C(n)/(n*ln(n))
1,14
1,12
1,1
1,08
1,06
1,04
1,02
1
0
20000
40000
60000
80000
100000
n

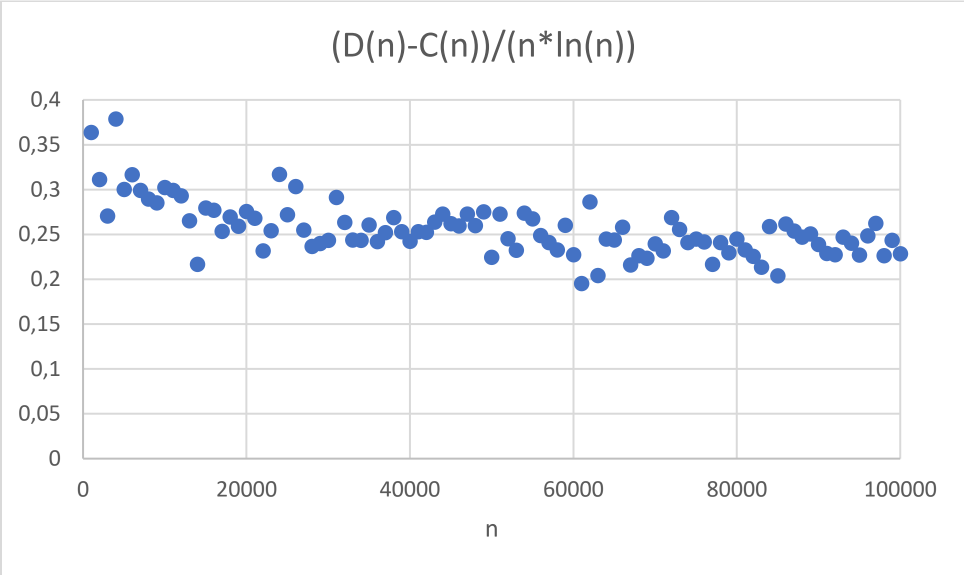

C(n)/n^2
0,008
0,007
0,006
0,005
0,004
0,003
0,002
0,001
0
0
20000
40000
60000
80000
100000
n

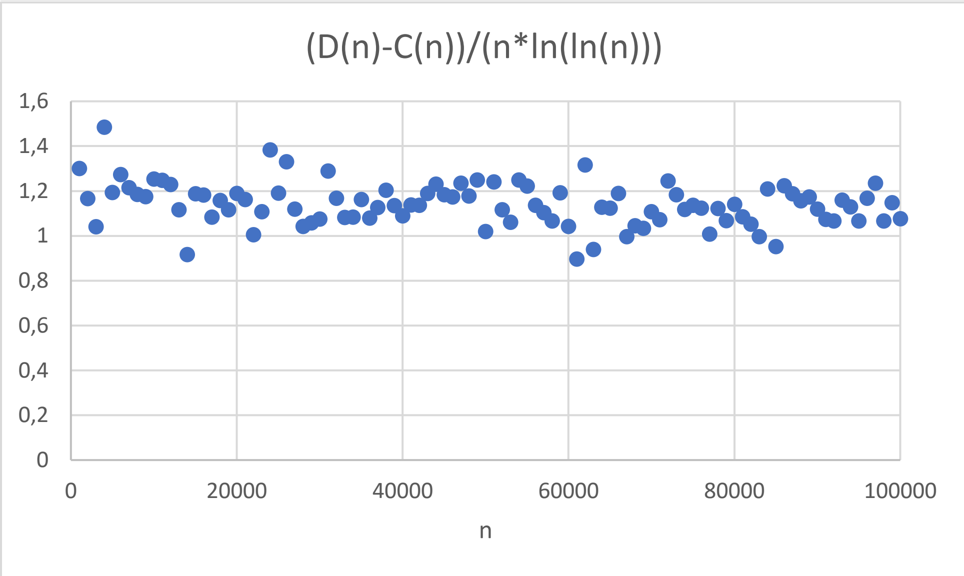

D(n)/n
16
14
12
10
8
6
4
2
0
0
20000
40000
60000
80000
100000
n


D(n)/(n*ln(n))
1,46
1,44
1,42
1,4
1,38
1,36
1,34
1,32
1,3
1,28
1,26
0
20000
40000
60000
80000
100000
n


D(n)/n^2
0,012
0,01
0,008
0,006
0,004
0,002
0
0
20000
40000
60000
80000
100000
n


(D(n)-C(n))/n
3,5
3
2,5
2
1,5
1
0,5
0
0
20000
40000
60000
80000
100000
n


(D(n)-C(n))/(n*ln(n))
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
20000
40000
60000
80000
100000
n


(D(n)-C(n))/(n*ln(ln(n)))
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
0
20000
40000
60000
80000
100000
n