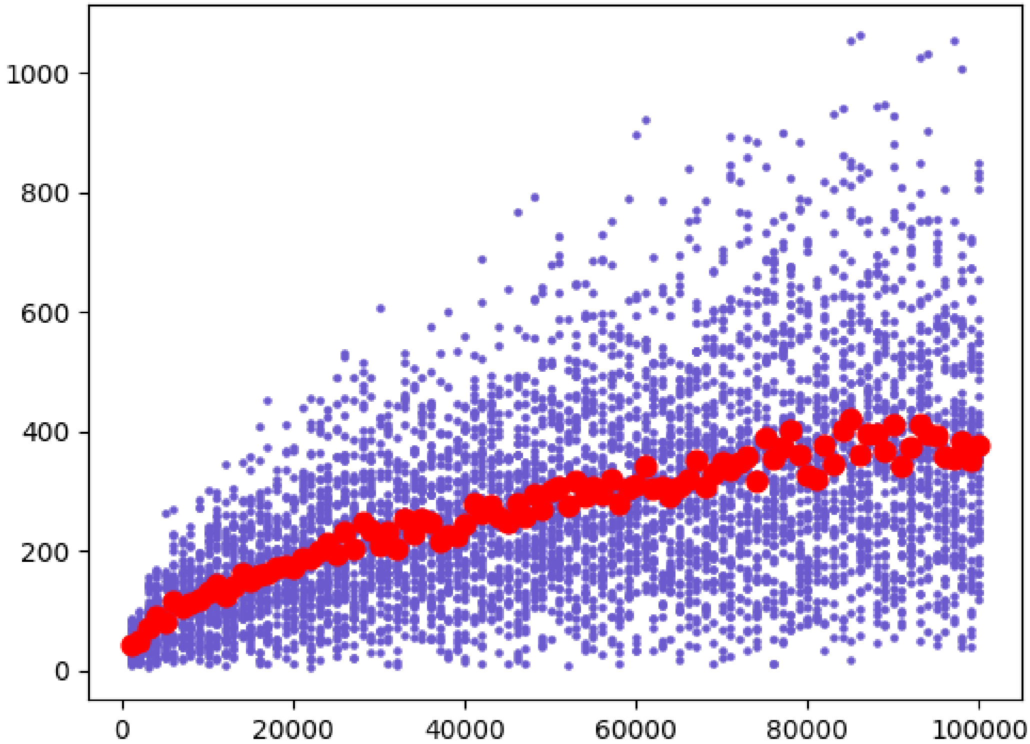
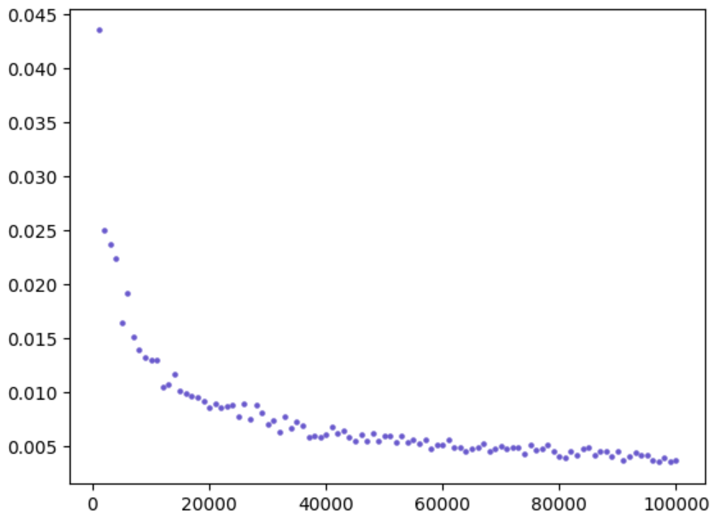
Metody Probabilistyczne i Statystyka
Zadanie Domowe 2

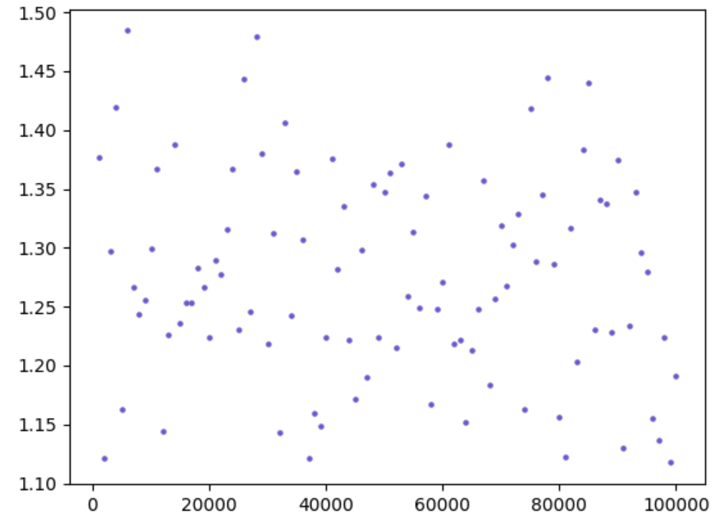
Autor: Dominik Gerlach



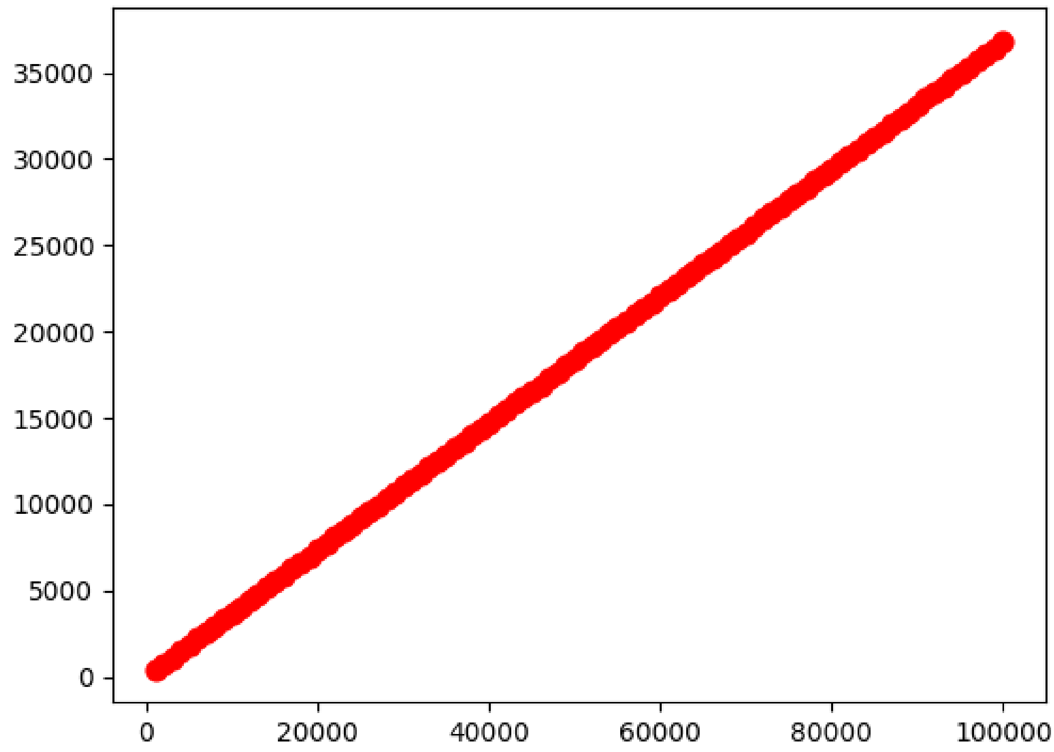
Rysunek 1: Wyniki eksperymentu dla B_n. Dla każdego n ϵ {1000, 2000, … ,100000} wykonano po k = 50 powtórzeń algorytmu. Niebieskie punkty odpowiadają wynikom z poszczególnych powtórzeń, czerwone punkty przedstawiają wartość średnią dla każdego n. Im większe n tym średnie odchylenie bezwzględne wyników z poszczególnych powtórzeń algorytmu jest większe. Dla konkretnego n wyniki z poszczególnych powtórzeń nie są silnie skoncentrowane wokół średniej wartości.



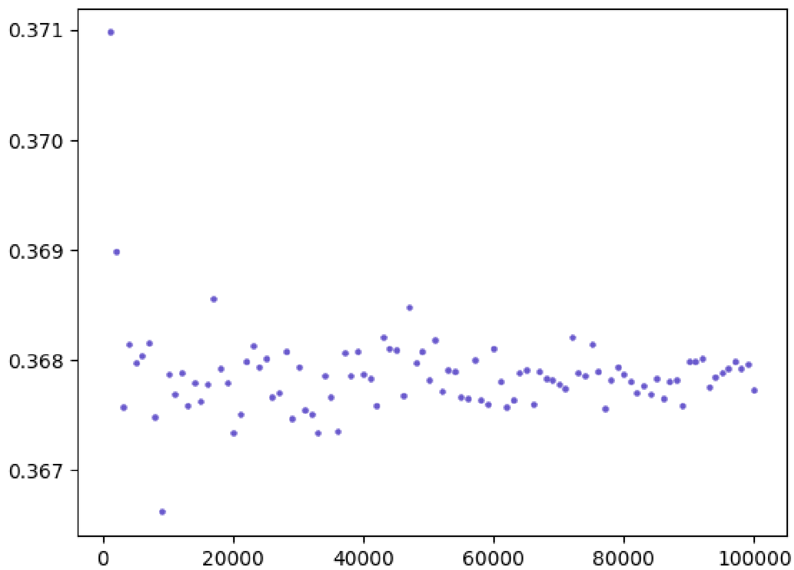
Rysunek 2: Przedstawia $\frac{b(n)}{n}$, gdzie b(n) odpowiada średniej wartości B_n dla danego n.



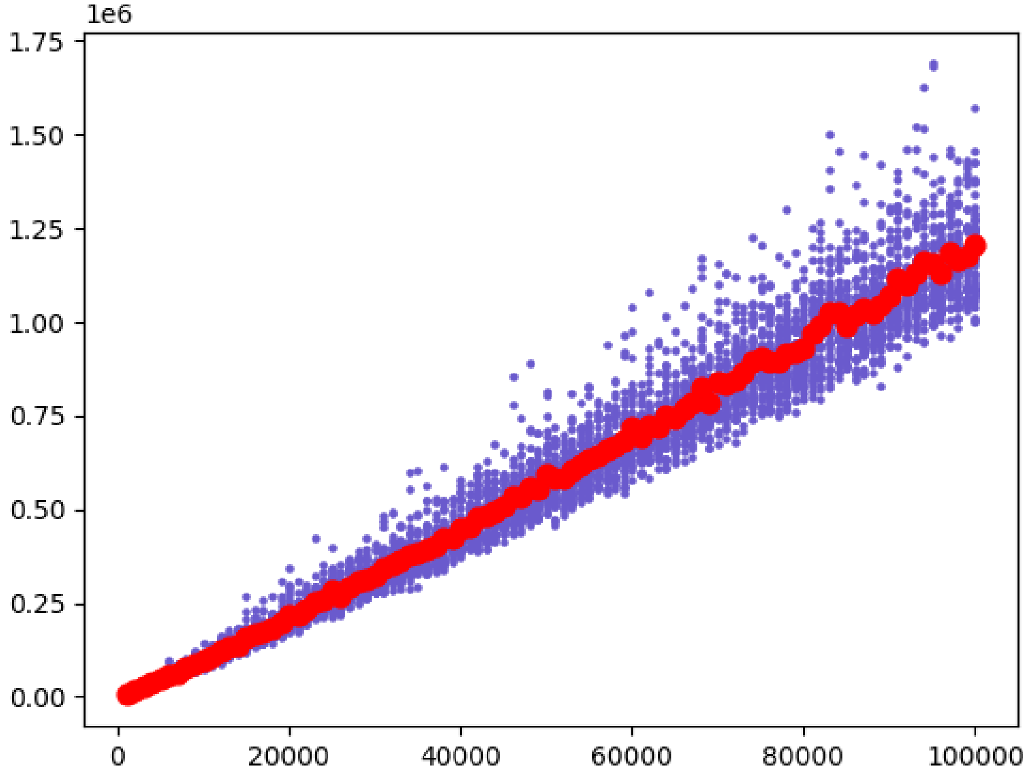
Rysunek 3: Przedstawia $\frac{b(n)}{\sqrt{n}}$, gdzie b(n) odpowiada średniej wartości B_n dla danego n.



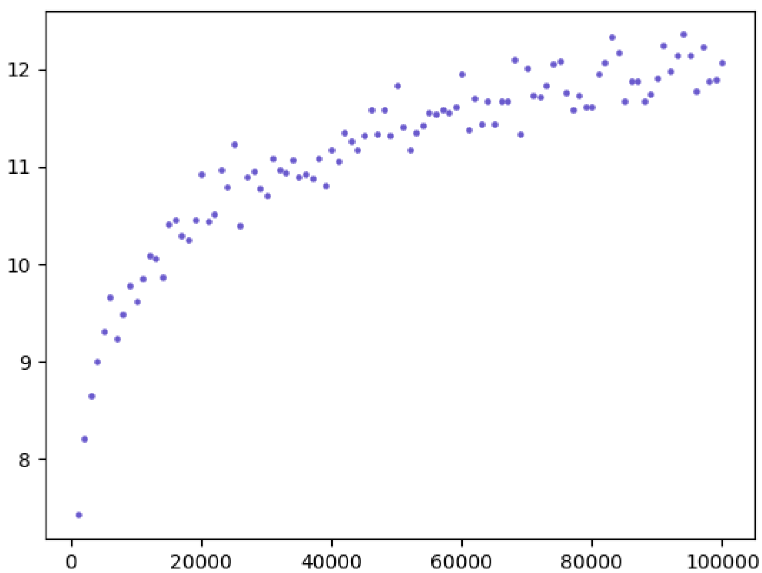
Rysunek 4: Rysunek 1: Wyniki eksperymentu dla U_n. Dla każdego n ϵ {1000, 2000, … ,100000} wykonano po k = 50 powtórzeń algorytmu. Niebieskie punkty odpowiadają wynikom z poszczególnych powtórzeń, czerwone punkty przedstawiają wartość średnią dla każdego n. Dla konkretnego n wyniki z poszczególnych powtórzeń są bardzo silnie skoncentrowane wokół średniej wartości.



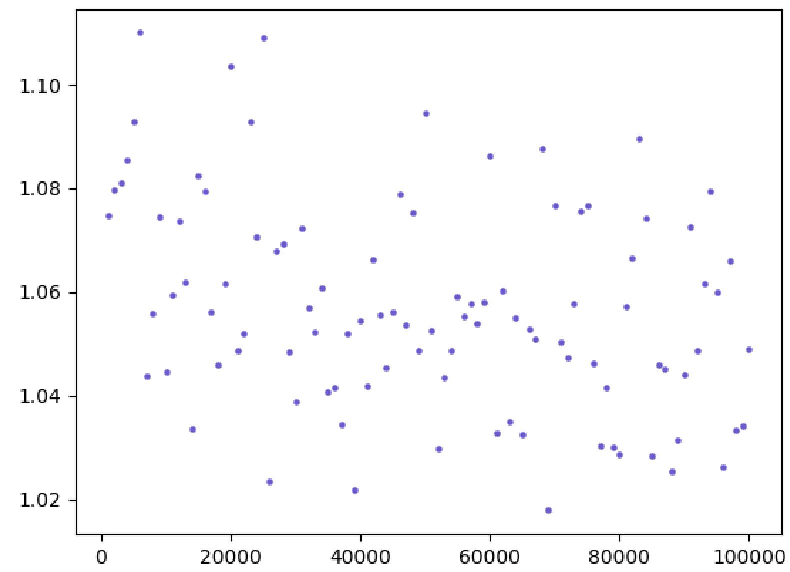
Rysunek 5: Przedstawia $\frac{u(n)}{n}$, gdzie u(n) odpowiada średniej wartości U_n dla danego n.



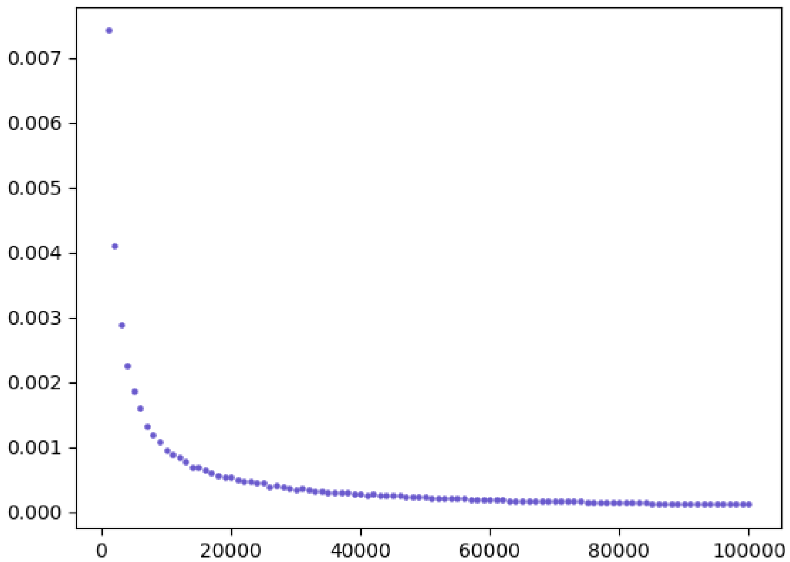
Rysunek 6: Rysunek 1: Wyniki eksperymentu dla C_n. Dla każdego $n \in \{1000, 2000, \dots, 100000\}$ wykonano po $k = 50$ powtórzeń algorytmu. Niebieskie punkty odpowiadają wynikom z poszczególnych powtórzeń, czerwone punkty przedstawiają wartość średnią dla każdego n. Im większe n tym średnie odchylenie bezwzględne wyników z poszczególnych powtórzeń algorytmu jest większe. Dla konkretnego n wyniki z poszczególnych powtórzeń są dosyć silnie skoncentrowane wokół średniej wartości.



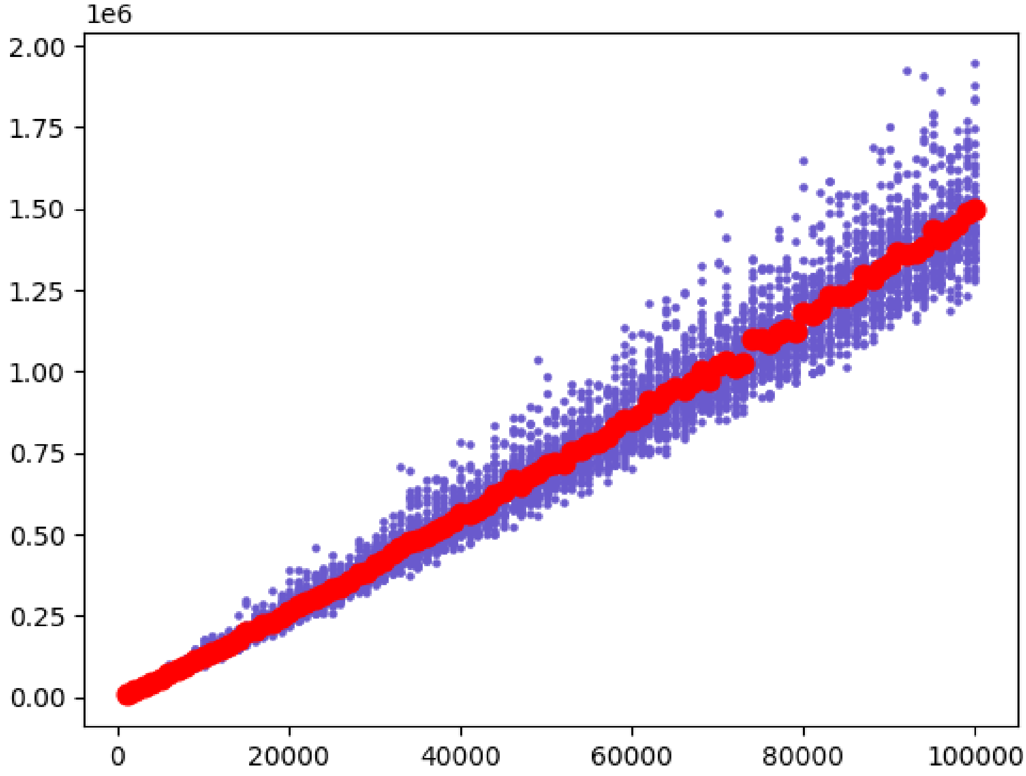
Rysunek 7: Przedstawia $\frac{c(n)}{n}$, gdzie c(n) odpowiada średniej wartości C_n dla danego n.



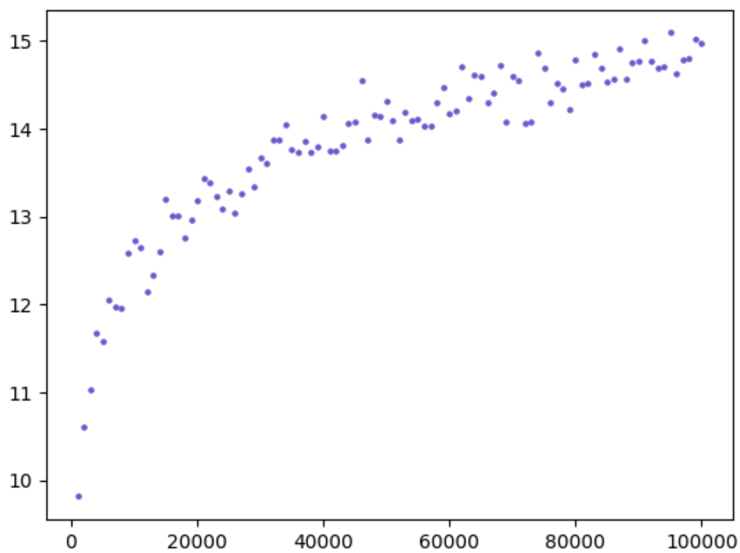
Rysunek 8: Przedstawia $\frac{c(n)}{n*ln(n)}$, gdzie c(n) odpowiada średniej wartości C_n dla danego n.



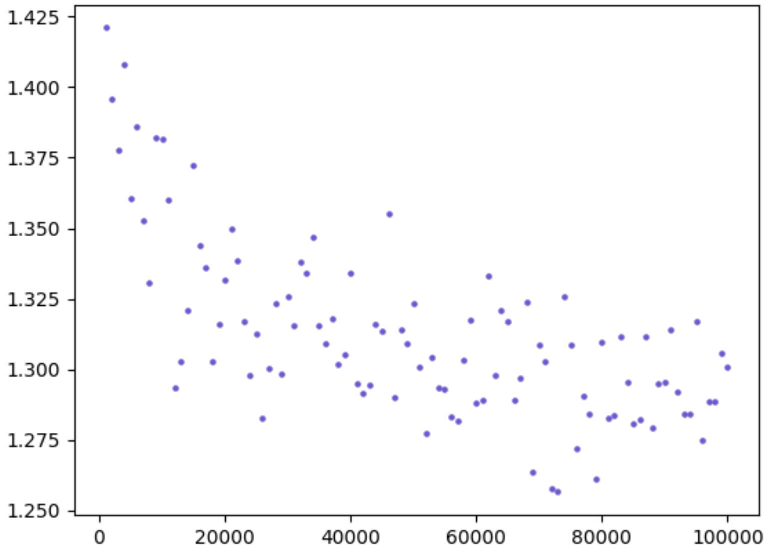
Rysunek 9: Przedstawia $\frac{c(n)}{n^2}$, gdzie c(n) odpowiada średniej wartości C_n dla danego n.



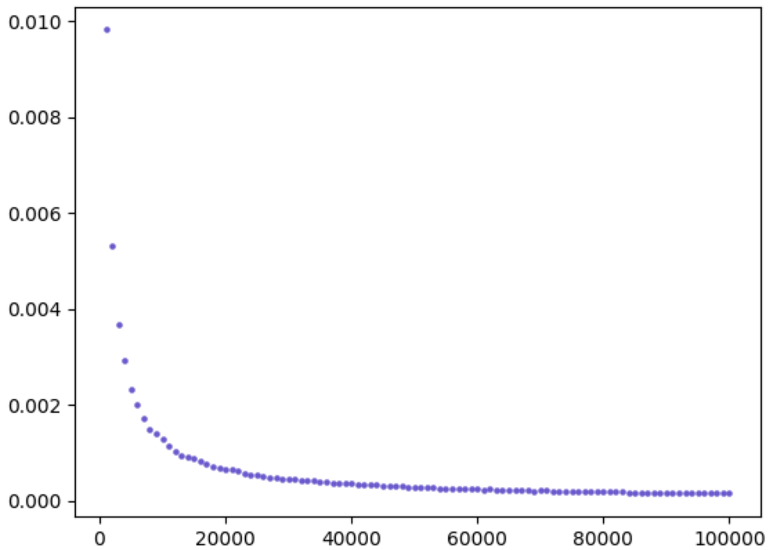
Rysunek 10: Rysunek 1: Wyniki eksperymentu dla D_n. Dla każdego n ϵ {1000, 2000, ... ,100000} wykonano po k = 50 powtórzeń algorytmu. Niebieskie punkty odpowiadają wynikom z poszczególnych powtórzeń, czerwone punkty przedstawiają wartość średnią dla każdego n. Im większe n tym średnie odchylenie bezwzględne wyników z poszczególnych powtórzeń algorytmu jest większe. Dla konkretnego n wyniki z poszczególnych powtórzeń są dosyć silnie skoncentrowane wokół średniej wartości.



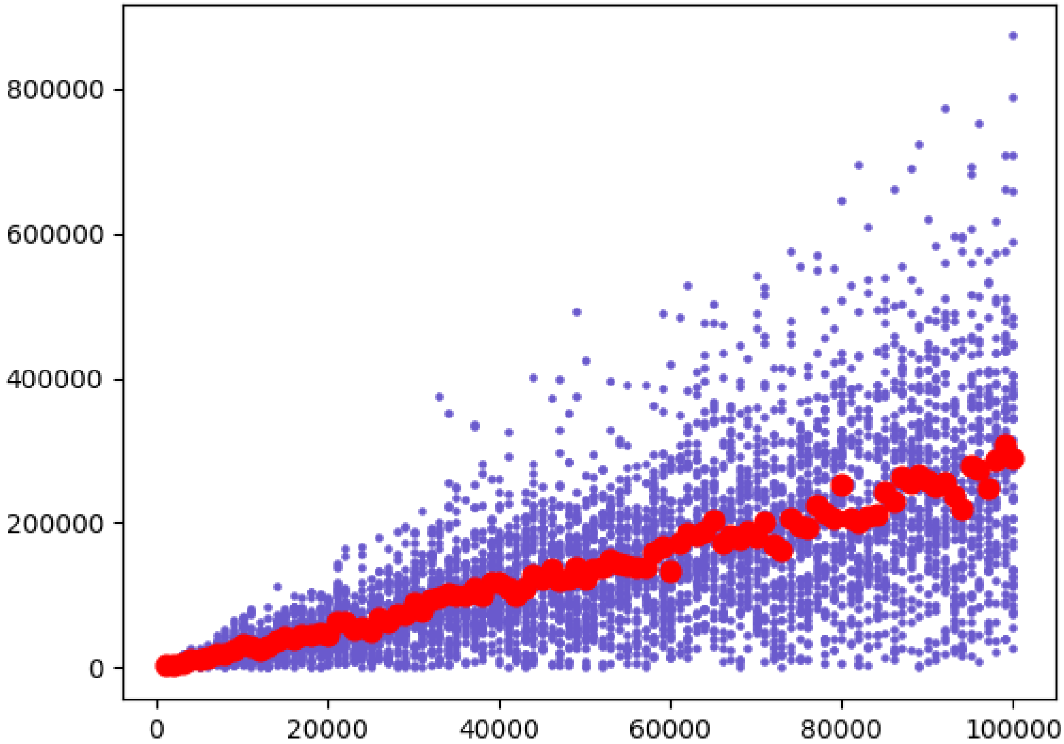
Rysunek 11: Przedstawia $\frac{d(n)}{n}$, gdzie d(n) odpowiada średniej wartości D_n dla danego n.



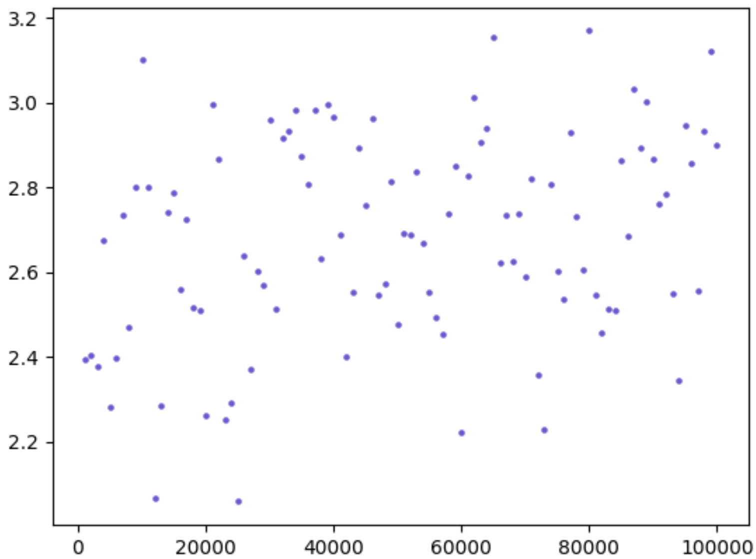
Rysunek 12: Przedstawia $\frac{d(n)}{n*ln(n)}$, gdzie d(n) odpowiada średniej wartości D_n dla danego n.



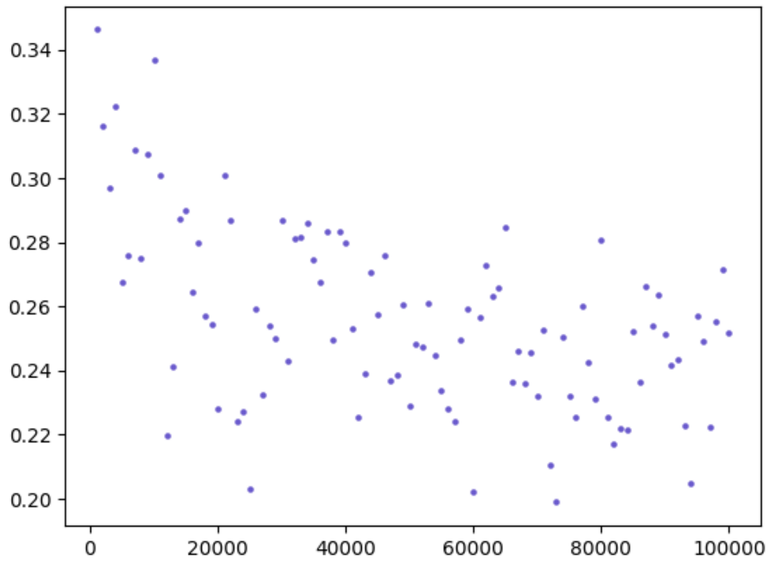
Rysunek 13: Przedstawia $\frac{d(n)}{n^2}$, gdzie d(n) odpowiada średniej wartości D_n dla danego n.



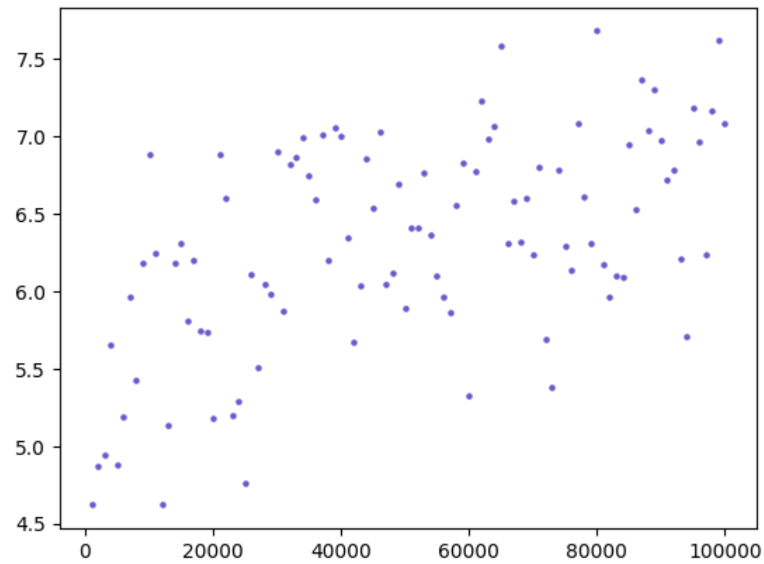
Rysunek 14: Rysunek 1: Wyniki eksperymentu dla D_n - C_n. Dla każdego n ϵ {1000, 2000, ... ,100000} wykonano po k = 50 powtórzeń algorytmu. Niebieskie punkty odpowiadają wynikom z poszczególnych powtórzeń, czerwone punkty przedstawiają wartość średnią dla każdego n. Im większe n tym średnie odchylenie bezwzględne wyników z poszczególnych powtórzeń algorytmu jest większe. Dla konkretnego n wyniki z poszczególnych powtórzeń nie są silnie skoncentrowane wokół średniej wartości.



Rysunek 5: Przedstawia $\frac{d(n)-c(n)}{n}$, gdzie d(n) - c(n) odpowiada średniej wartości D_n - C_n dla danego n.



Rysunek 6: Przedstawia $\frac{d(n) - c(n}{n^{*}ln(n)}$, gdzie d(n) - c(n) odpowiada średniej wartości D_n - $C_n$dla danego n.



Rysunek 7: Przedstawia $\frac{d(n) - c(n)}{n^{*}ln(ln(n))}$, gdzie d(n) - c(n) odpowiada średniej wartości D_n - C_n dla danego n.

Definicje:

- **Bn** – moment pierwszej kolizji; Bn = k, jeśli k-ta z wrzucanych kul jest pierwsza, która trafiła do niepustej urny,
- **Un** – liczba pustych urn po wrzuceniu n kul,
- **Cn** – minimalna liczba rzutów, po której w każdej z urn jest co najmniej jedna kula (pierwszy moment, w którym nie ma już pustych urn; problem kolekcjonera kuponów, ang. coupon collector's problem),
- **Dn** – minimalna liczba rzutów, po której w każdej z urn są co najmniej dwie kule,
- **Dn − Cn** – liczba rzutów od momentu Cn potrzeba do tego, żeby w każdej urnie były co najmniej dwie kule.

Kryteria: Hipotezę odnośnie asymptotyki wartości średnich badanych wielkości formułuję na podstawie wykresów $\frac{F}{L}$, gdzie F ϵ {b(n), u(n), c(n), d(n), d(n) - c(n)} i L ϵ {n, n^2, $\sqrt{n}$, n*ln(n), n*ln(ln(n)) }. Im wykres bardziej odpowiada funkcji stałej, tym lepiej opisuje tempo wzrostu funkcji F.

Hipotezy: Korzystając z powyższych kryteriów formułuję następujące hipotezy:

- b(n) = O($\sqrt{n}$)
- u(n) = O(n)
- c(n) = O(n * ln(n))
- d(n) = O(n * ln(n))

- Ciężko stwierdzić na podstawie badanych wykresów asymptotykę funkcji d(n) - c(n). Biorąc pod uwagę asymptotyki d(n) i c(n) możemy posunąć się do stwierdzenia, że d(n) - c(n) = O(n * ln(n))

Paradoks urodzinowy:
Załóżmy, że mamy 365 kartonów, kartony po kolei oznaczają dzień z roku nieprzestępnego. Następnie z grupy osób, losowo wybieramy osoby i nakazujemy im wejście do kartonu z dniem ich urodzin. Ile wybranych osób nam potrzeba, żeby zaszła pierwsza sytuacja, w której karton będą zajmowały dwie osoby? Odpowiedź jest nieintuicyjnie mała, z tego powodu problem ten nazwany jest paradoksem urodzin. Symulację dla tego przypadku można wykonać korzystając z algorytmu obliczającego B_{365}.

Problem kolekcjonera kuponów:
Kolekcjoner kart ma album, który może pomieścić n kart. Nasz bohater codziennie kupuje jedną losową kartę, której jest przypisane miejsce w albumie. Ile kart musi kupić kolekcjoner, by album został zapełniony wszystkimi kartami? Symulację dla tego problemu można wykonać korzystając z algorytmu obliczającego Cn.

Jakie znaczenie ma paradoks urodzin w kontekście funkcji haszujących i kryptograficznych funkcji hashujcych?
Paradoks urodzin w funkcjach haszujących jest przydatny przy pytaniu: jak wiele wyników funkcji haszujących musi zostać wygenerowanych, by zaszła kolizja, czyli sytuacja, w której inne argumenty funkcji haszującej dają tę samą wartość.

Istotne jest więc projektowanie odpornych na kolizje funkcji haszujących, by przeciwdziałać potencjalnym “atakom urodzinowym”.
