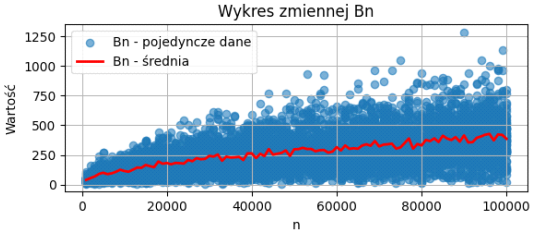
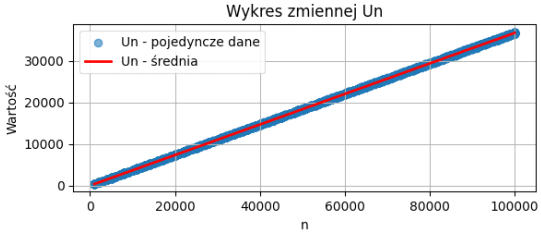
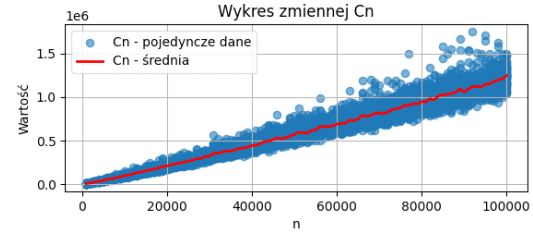
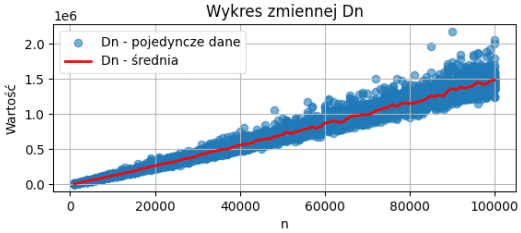
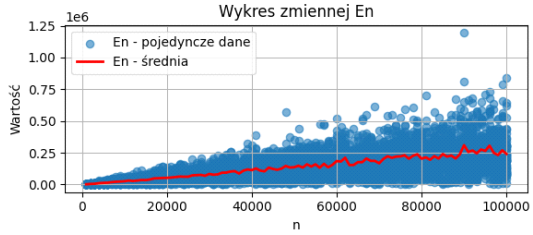


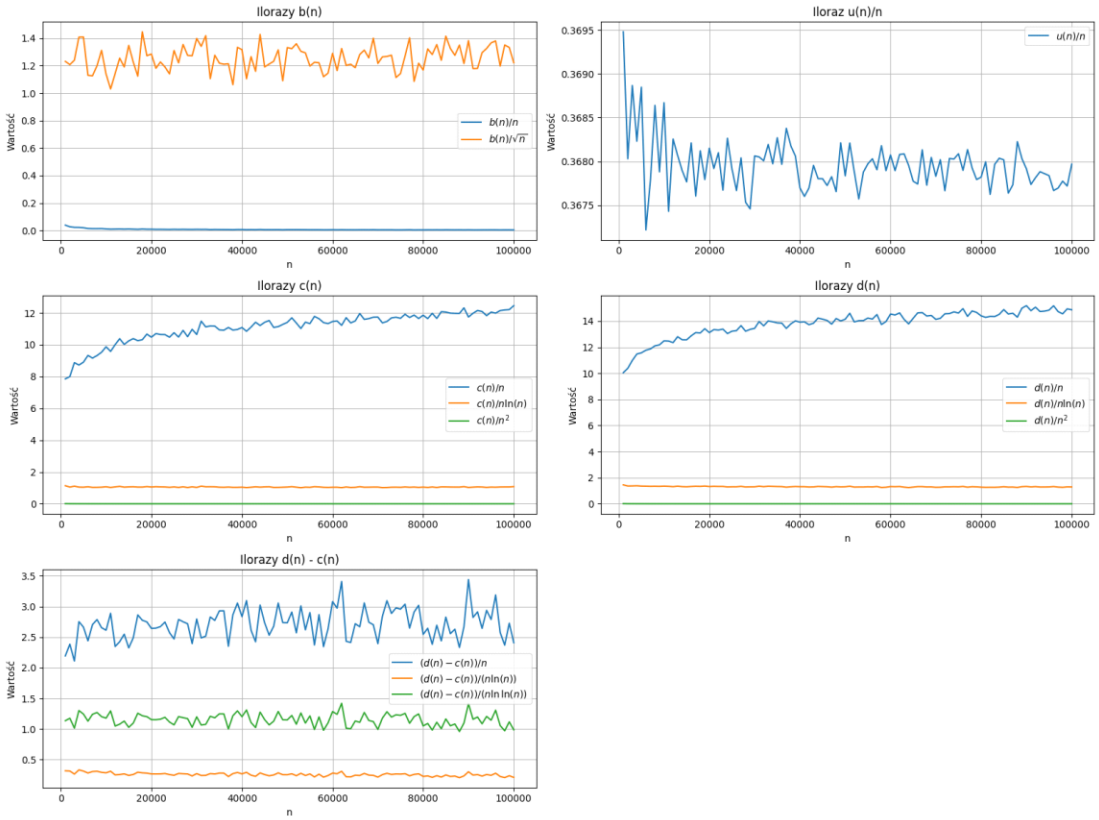








W przypadku każdej funkcji pojedyńcze dane wskzują tendencję do wzrostu wraz ze wzrostem n, czego można się spodziewać , ponieważ zwiększania n zwiększa zakres zdarzeń w badanych procesach (poza En oczywiście, gdzie sytuacja nie jest taka oczywista).



Przedstawione wyżej ilorazy funkcji pozwalają oszacować asymptotyczne tempo wzrtosu funkcji b(n), c(n), d(n), e(n), u(n) a są to kolejno **sqrt(n)**, **n**, **n*ln(n)**, **n*ln(n)**, **n*ln(ln(n)**

(***)

b)

Wykresy pokazują, że wyniki poszczególnych powtórzeń są zbliżone do wartości średnich dla małych wartości. Im większe n, tym większe są względne odchylenia od wartości średniej, co wskazuje na brak stabilności asymptotycznej.

En, jako różnica dwóch rosnących funkcji, ma nieco większe rozrzuty już dla małych n.\

c) Zostało omówione wyżej przy omawianiu wykresu. Patrz(***)

Wyjaśnienie skąd biorą się poszczególne asymptotyki:

- Bn: Średnia wartość rośnie asymptotycznie jak pierwiastek z n, co widać z proporcji b(n)n/sqrt(n), która stabilizuje się dla dużych n.

- Un: Średnia wartość jest proporcjonalna do n, ponieważ u(n)/n zbliża się do stałej.
- Cn: Wartość średnia rośnie asymptotycznie jak n*ln(n). Wykres c(n)/n*ln(n) wskazuje na stabilność tej proporcji.
- Dn: Podobnie jak Cn, średnia wartość Dn rośnie asymptotycznie jak n*ln(n), co widać z analogicznych proporcji.
- Dn−Cn: Różnica ta rośnie wolniej niż n*ln(n), ale szybciej niż n, co sugeruje asymptotykę w postaci n*ln(ln(n)).

d)

- Birthday paradox - paradoks ten polega na tym, że intuicja podpowiada, że skoro rok ma 365 dni potrzebujemy wielu osób (mi osobiście podpowiadała, że w okolicach 100), żeby natrafić na dwie osoby o tej samej dacie urodzin (nie uwzględniając roku). W rzeczywistości okazuje się, że potrzeba średnio sqrt(365), co wynosi w przybliżeniu 19 osób, a będzie wsród nich para osób, która ma urodziny tego samego dnia i miesiąca. Dokładnie to samo dzieje się, gdy wybieramy losowo urny, do których wrzucamy kule aż do momentu, gdy natrafimy na urne, do której już była wrzucona jedna kula.
- Coupon collector's problem - problem ten polega dokładnie na tym, co liczymy w podpunkcie w podpunkcie c pierwszego zadania. Chcemy policzyć liczbę wrzuceń kul do urn, tak, żeby w każdej urnie była przynajmniej jedna kula, zakładając, że wybór urny jest losowo (jednorodny rozkład prawdopodobieństwa). Intuicja jest taka, że początkowo łatwo "zbierać kolejne kupony", jako że urny są puste, ale im dalej, tym ciężej trafić na pustą urnę, bo wiekszość z nich już jest zapełniona przynajmniej jedną kulą.

e)

- "Birthday paradox" tłumaczy, dlaczego zderzenia (kolizje) w funkcjach haszujących są bardziej prawdopodobne niż się wydaje. Dla przestrzeni o rozmiarze n, wystarczy średnio sqrt(n) losowych wartości, aby wywołać kolizję.
- Kryptograficzne funkcje haszujące muszą być odporne na ataki typu "birthday attack", w których haker próbuje wygenerować dwie różne wiadomości z takim samym haszem. Dlatego rozmiar haszy (np. 256 bitów) jest wybierany wystarczająco duży, aby sqrt(n) było trudne do osiągnięcia w praktyce.

Karol Wziątek, 279734