# Analiza Algorytmów, Lista 2 Raport

Tomasz Krent

April 2, 2020

## 1 Wstep

Programy zostały napisane w jezyku python. Algorytmy $MinCount$ oraz $HyperLogLog$, których dotycza odpowiednie zadania znajduja sie odpowiednio w plikach $minCount.py$ oraz $hll.py$.

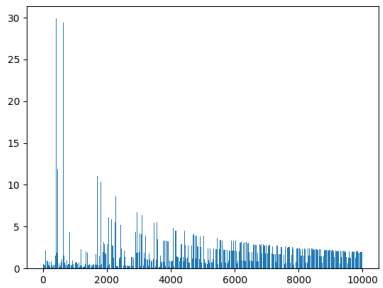
## 2 Zadanie 1

(a) Wyniki działania programu $zad1a.py$ dla $n = 1000$, $k = 400$ oraz dla różnych parametrów $l$ oznaczajacych ilość powtórzonych elementów w multizbiorze:

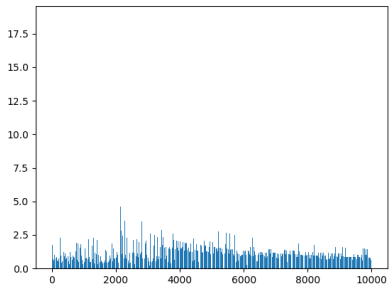
- dla $l = 0$ $E[X] = 1006.3572237833873$ i $Var[X] = 1624.0635753305821$
- dla $l = 100$ $E[X] = 893.0812380685884$ i $Var[X] = 855.8069043679233$
- dla $l = 200$ $E[X] = 790.5833137404738$ i $Var[X] = 704.1158101740564$
- dla $l = 500$ $E[X] = 497.90574849334297$ i $Var[X] = 103.5780509228641$

Widać tutaj, że ilość powtórzeń w multizbiorze ma wpływ na wartość oczekiwana i wariancje, które odpowiednio maleja wraz ze wzrostem powtórzonych elementów.

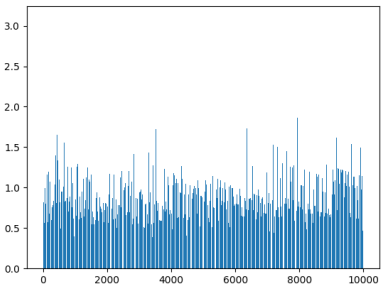
(b) Wyniki działania programu $zad1b.py$ w postaci wykresów słupkowych dla $n = 1..10000$ oraz dla różnych $k$ sprawdzajacego jak współczynnik $k$ wpływa na wyniki działania algorytmu:



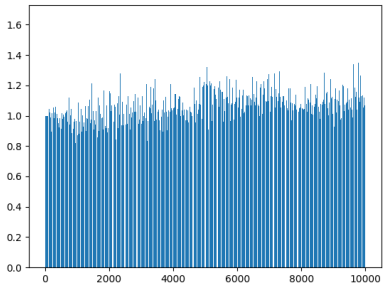

(a) k=2



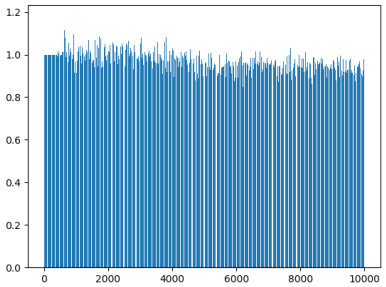

(b) k=3




(c) k=10




(a) k=100




(b) k=400

Widać, że wraz ze wzrostem parametru $k$ stosunek $\tilde{n}/n$ do $n$ jest coraz bliższy jedynce, przez co wyniki działania algorytmu sa lepsze.

(c) Wyniki działania programu *zad1c.py* w postaci tabeli dla $n = 1000$, $k = 10, 20, 30..1000$, ukazujacej w ilu przypadkach na 1000 dla danego $k$ $|\tilde{n}/n - 1| < 0.1$:

- [ 361, 331, 414, 514, 610, 573, 681, 722, 694, 653, 732, 727, 760, 817, 837, 835, 865, 876, 902, 901, 926, 919, 931, 937, 933, 941, 943, 952, 961, 948, 962, 987, 986, 982, 989, 985, 990, 993, 994, 998, 994, 996, 998, 997, 997, 996, 998, 1000, 999, 999, 1000, 998, 1000, ... ,1000 ]

Widzimy tutaj, że wraz ze wzrostem $k$ w coraz wiekszej ilości przypadków. $|\tilde{n}/n - 1| < 0.1$. Widać również, że wartość 95% na 1000 prób została przekroczona dla $k$ wynoszacego około 400.

.

# 3 Zadanie 2

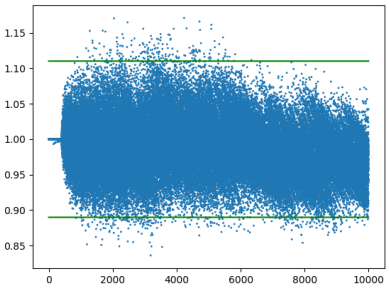
Wyniki działania programu *zad2.py* dla $n = 1000$ oraz $k = 400$ na podstawie 1000 prób, dla różnych parametrów $B$, dla funkcji haszujacej: $h \to \{0, 1\}^B$:

- dla $B = 40$ $E[X] = 9827.168549489272$
- dla $B = 32$ $E[X] = 9903.36637669319$
- dla $B = 24$ $E[X] = 9891.689089620297$
- dla $B = 20$ $E[X] = 10116.185213417126$
- dla $B = 18$ $E[X] = 9168.149010952331$
- dla $B = 16$ $E[X] = 8581.130220868841$
- dla $B = 14$ $E[X] = 7584.260926382609$
- dla $B = 12$ $E[X] = 3728.9435903850263$

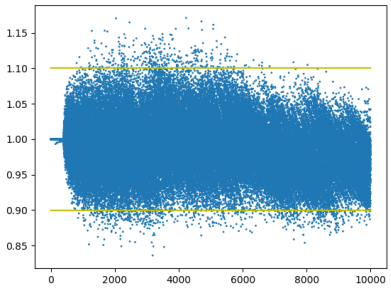
Wraz ze zmniejszaniem parametru $B$ wyniki algorytmu sa zdecydowanie gorsze. Spowodowane jest to tym, że hasze sie powtarzaja i dochodzi do kolizji powodujac obniżenie wartości końcowej $\tilde{n}$.

# 4 Zadanie 3

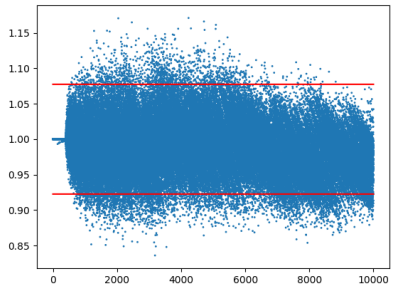
Wyniki programu *zad3b.py* dla $n = 1..10000$, $k = 400$, $\alpha = 5\%, 1\%i, 0.5\%$ oraz 10 prób dla każdego $n$ w postaci wykresów przedstawiajacych wartość $\tilde{n}/n$ oraz wartości $1 - \delta$ oraz $1 + \delta$ takich że: $\Pr[1\text{-}\delta < \tilde{n}/n < 1 + \delta] > 1 - \alpha$



(a) $\alpha = 0.5\%, \delta \approx 0.110229$

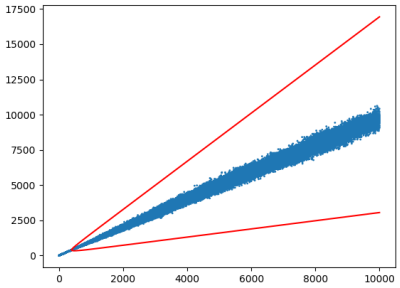


(b) $\alpha = 1\%, \delta \approx 0.100528$



(c) $\alpha = 5\%, \delta \approx 0.077446$

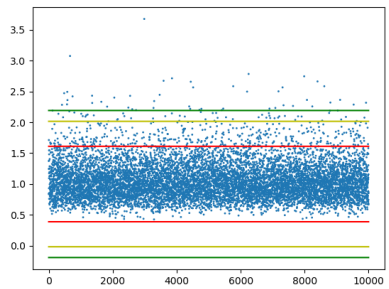
Teoretyczne wyniki uzyskane przez nierówność Czebyszewa wzgledem wyników działania programu *zad3.py* oraz *zad3a.py*:



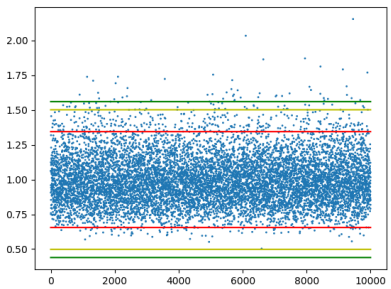
Widać tutaj, że ograniczenia nie sa w tym przypadku zbyt precyzyjne.

# 5 Zadanie 4

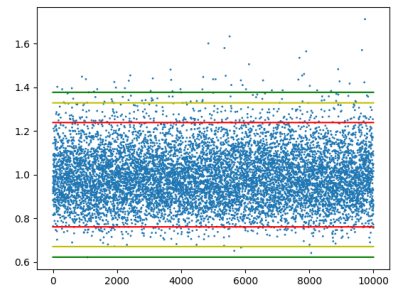
Wyniki działania programów *zad4_hll.py* oraz *zad4_wykres* sprawdzajacego działanie algorytmu *HiperLogLog* dla różnych wartości parametru $m = 16, 32, 64$ oraz $n = 1000$ dla 10000 prób w postaci $\tilde{n}/n$ dla każdej próby. Na tych wykresach również zostały zaznaczone linie oznaczajace tak jak w zadaniu poprzednim odpowiednie linie $1 - /delta$ oraz $1 - /delta$ dla odpowiedniego $/alpha = 0.05, 0.01, 0.005$ takich, że $\Pr[1\text{-}\delta < \tilde{n}/n < 1 + \delta] > 1 - \alpha$



(a) m=16



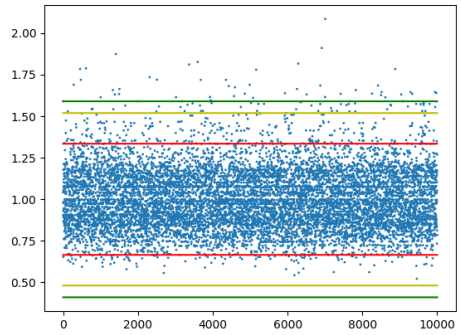
(b) m=32



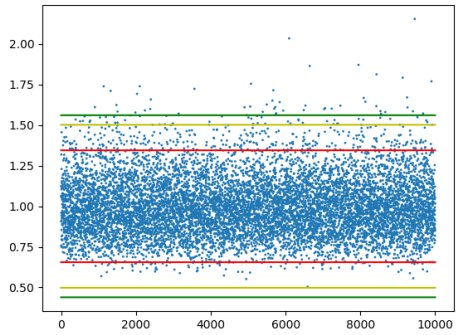
(c) m=64

Z wykresów można wywnioskować, że wraz ze wzrostem parametru $m$ zwiekszała sie dokładność uzyskanych wyników działania programu.

Kolejne wykresy uzyskane przy pomocy programów: *zad4_mc.py*, *zad4_hll.py* oraz *zad4_wykres* narysowane w taki sam sposób jak poprzednie przedstawiaja wyniki uzyskane przez algorytmy $MinCount$ oraz $HyperLogLog$, w sytuacji gdy oba maja do dyspozycji te sama ilość pamieci:




(a) wyniki dla $MinCount$




(b) wyniki dla $HyperLogLog$

Wyniki obu algorytmów sa bardzo podobne, linie przedstawiajace $\delta$ dla $\alpha = 5\%, 1\%, 0.5\%$ znajduja sie mniej wiecej na tej samej wysokości na obu wykresach.