Sprawozdanie AAL

Tablica mieszająca **w12w21w31**

Maciej Kasprzyk

1. Treść zadania

w12w21w31

Przedmiotem analizy jest tablica mieszająca: tablica przechowuje rekordy zawierające napisy. Długość tablicy jest ograniczona arbitralnie przez pewną stałą K. Dla danego napisu s obliczamy k=M(s) i umieszczamy strukturę reprezentującą napis w tablicy mieszającej: H[k]. W przypadku kolizji funkcji mieszającej (H[k] zajęte) reprezentujące napis s struktury danych zapisywane są w sposób alternatywny zobacz warianty). Przedmiotem implementacji powinno być: dodanie i usunięcie elementów w H[]. Wybór funkcji mieszającej M(s) do decyzji projektanta - ale patrz wariant 3.

W12 W przypadku kolizji funkcji mieszającej reprezentujące napis s struktury danych zapisywane są w liście jednokierunkowej, której głowa to H[k].

W21 Testy przeprowadzić dla: listy słów języka polskiego wygenerowanych z zadanych tekstów. Generator należy wykonać samodzielnie.

W31 Zastosować jedną funkcję mieszającą; dodatkowo przeprowadzić analizę dla enumeracji tablicy (wydobycia wszystkich elementów).

2. Opis zrealizowanego zadania

2.1. Tablica mieszająca

Została zaimplementowana tablica mieszająca w języku C++. Możliwe jest dodawanie i usuwanie elementów ze struktury. Dodatkowo można enumerować po elementach tablicy z użyciem zaimplementowanego iteratora. W przypadku kolizji elementy przechowywane są na liście jednokierunkowej, została użyta struktura *forward_list* ze standardowej biblioteki.

Tablica mieszająca zabezpieczona jest przed przypadkami niedozwolonego użycia, w przypadku próby usunięcia lub odwołania się do nieistniejącego elementu rzucany jest wyjątek.

2.2. Funkcja mieszająca

Słowa języka polskiego są przechowywane w kodowanu UTF-8 zarówno w pliku jak w obrębie programu C++. W związku z tym, że polskie znaki kodowane są za pomocą kilku bajtów, pętla funkcji mieszającej wykona do nich kilka obrotów. Nie wpływa to na jej poprawność.

Postać funkcji mieszającej:

```
int hash = 0;
for (int i = 0; i < długość_w_bajtach; i++) {
  hash = (hash * 31 + wartość(i) ) % wielkość_tablicy;
}
```

2.3. Pomiar czasu

Do pomiaru czasu służy program *main.cpp*, który do określonej wielkości tablicy oraz ilości elementów mierzy czas wykonywania operacji. Najpierw dodawana jest określona ilość elementów do tablicy, później mierzony jest czas odpowiednio usunięcia, dodania oraz enumeracji po wszystkich elementach. Wyniki wypisywane są na standardowe wyjście. Pomiar czasu został wykonany z użyciem standardowej biblioteki *chrono.* Skrypt *analizator.py* uruchamia wielokrotnie program *main.cpp* dla coraz większej ilości elementów. Przyjęto zakres od 1/100 n do n. Skrypt prezentuje wyniki w postaci wykresów oraz tabeli.

2.4. Generowanie słów języka polskiego

Do generowanie słów języka polskiego wykorzystano język Python. Na początku pobierane są teksty wszystkich dostępnych lektur polskich z serwisu https://wolnelektury.pl. Generator:

1) usuwa z początku i końca słowa znaki, które nie są literami
2) ignoruje słowa w dalszym ciągu zawierające znaki, które nie są literami
3) pomija słowa krótsze niż 3 litery
4) zapewnia unikalność słów

Udało się wygenerować ponad pół miliona różnych słów. Po zapoznaniu się z częścią wyników odnosi się wrażenie, że zdecydowana większość słów to faktycznie poprawne polskie słowa, jednak pojawiają się słowa z innych języków. Z uwagi na obszerność danych nie przeprowadzono dalszej filtracji słów.

2.5. Uwaga odnośnie implementacji

W celu wykonania wielu pomiarów dla każdej kombinacji parametrów tworzony jest nowy proces, na którym wykonywany jest program *main.cpp.* W trakcie realizacji okazało się, że dla rzetelności otrzymanych wyników trzeba przeprowadzić więcej pomiarów niż początkowo zakładano (rzędu kilkuset). Doprowadziło to do tego, że w tworzona i niszczona jest duża ilość pojedynczych procesów co spowalnia program. Mimo tego spowolnienie było mało znaczące i czas oczekiwania na wyniki był rozsądny.

3. Podział na pliki

hashmap.h - implementacja hashmapy
main.cpp - pomiar czasu
analizator.py - prezentacja wyników w postaci wykresów oraz tabeli
data_ download.py - pobieranie danych książek
download_ menu.py - interfejs tekstowy do pobierania książek
generator.py - parsowanie pobranych tekstów i generowanie słów języka polskiego
test.cpp - test poprawności działania hashmapy

4. Opis funkcjonalny

Bezpośrednio przez użytkownika wykorzystywane są trzy moduły:

1) analizator.py *słowa folder n k*
   słowa - ścieżka do pliku txt zawierającego słowa, które mają być użyte do testowania
   folder - folder, w którym zostaną zapisane wykresy i tabela
   n - wielkość tablicy używana strukturze
   k - maksymalna ilość wstawionych elementów, dla których zostanie wykonany pomiar

2) generator.py [-*n ilość*] *plik*
   ilość - liczba słów do wygenerowania, jeżeli nie podany wygenerowane zostanie maksimum
   plik - plik zawierający teksty książek w formacie JSON
3) download_menu.py
   Po uruchomieniu interfejs tekstowy do pobierania z API.

5. Wersja Windows

Kod pisany w ramach projektu był tworzony z myślą o przenośności, dlatego wersja windows potrzebowała tylko niewielkich, mało znaczących zmian.

6. Testowanie

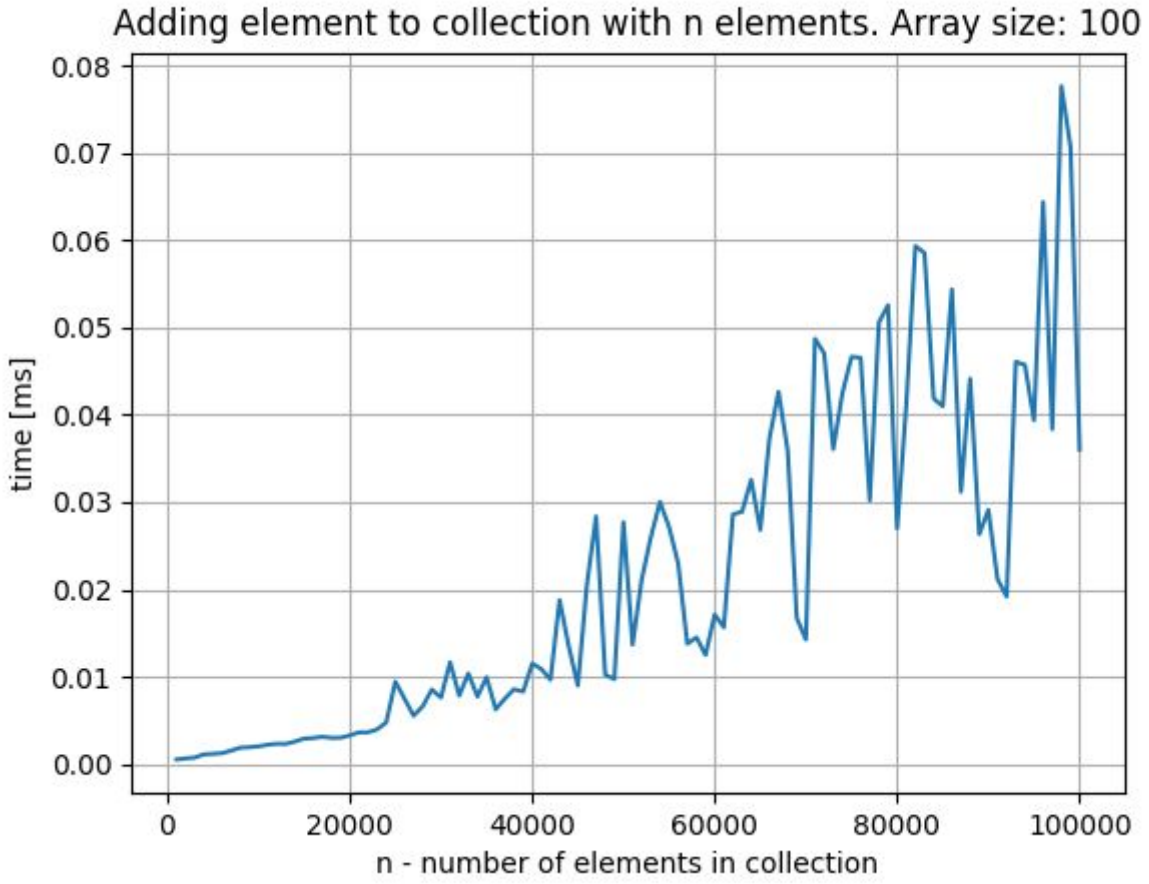
Przetestowano działanie dodawania, usuwania i enumeracji elementów. Testy znajdują się w pliku *test.cpp.*

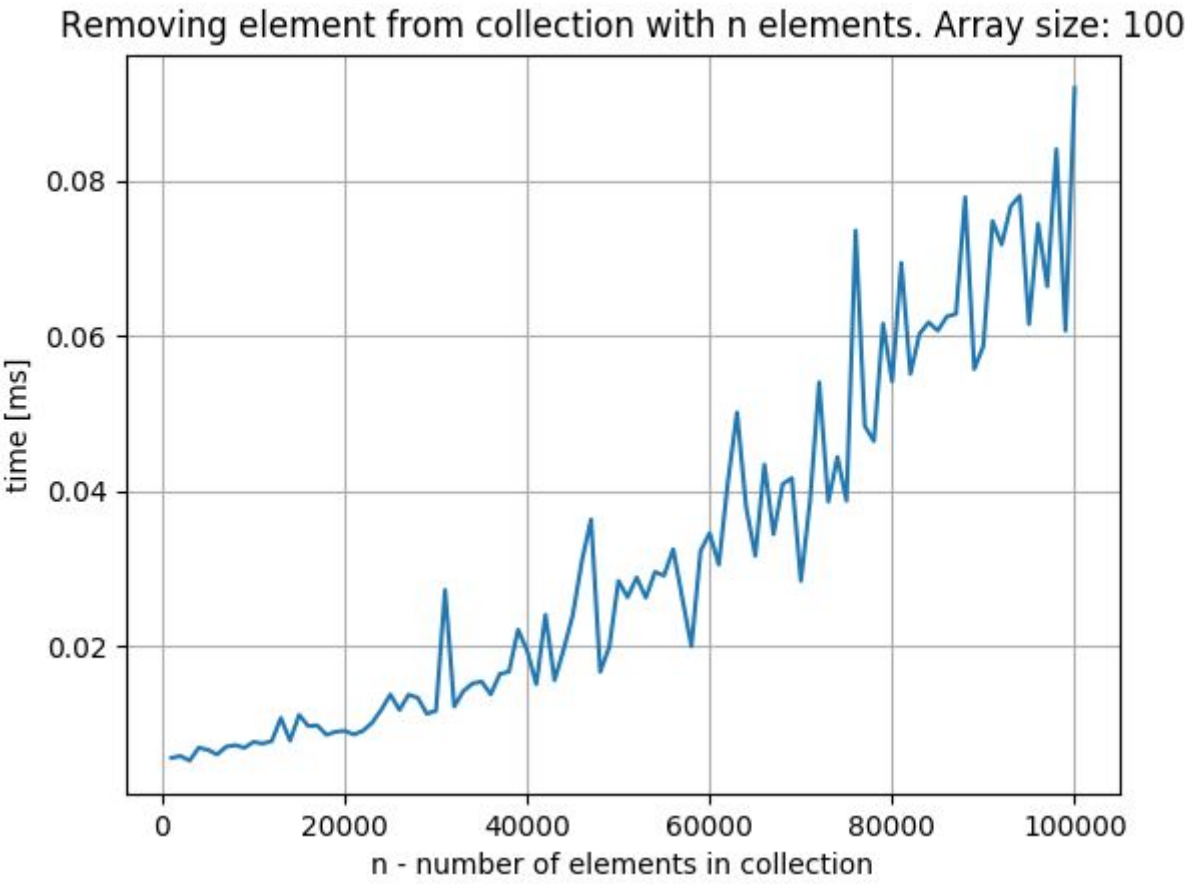
7. Wyniki pomiarów i wnioski

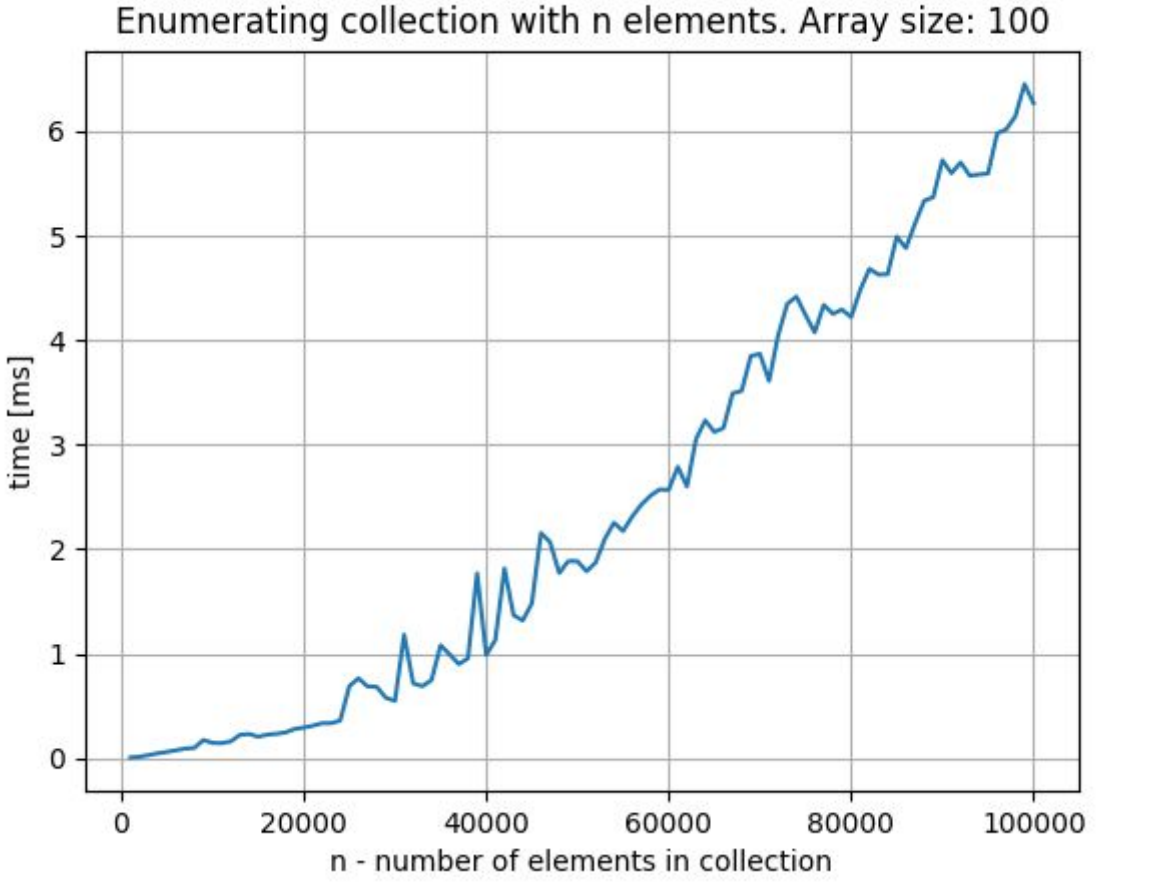
Przy obliczeniu wartości parametru **q(n)** założony czas stały dla operacji dodawania i usuwania oraz liniowy dla operacji enumeracji. Każdy pomiar wykonano pięć razy i jako wynik przyjęto wartość średnią. Czas podano w ms.

7.1. Przykładowe uzyskane wyniki

7.1.1. Rozmiar tablicy: 100, maksymalna ilość elementów 100 000




Removing element from collection with n elements. Array size: 100
time [ms]
0.02
0.04
0.06
0.08
0
20000
40000
60000
80000
100000
n - number of elements in collection


Enumerating collection with n elements. Array size: 100
time [ms]
0
1
2
3
4
5
6
0
20000
40000
60000
80000
100000
n - number of elements in collection

Remove

n	t(n)	q(n)
10000	0.0076424	0.221223
20000	0.009079	0.262807
30000	0.0116608	0.337542
40000	0.0194008	0.56159
50000	0.0283978	0.822024
60000	0.0345462	1
70000	0.0284014	0.822128
80000	0.0540924	1.5658
90000	0.0586856	1.69876
100000	0.0918934	2.66001

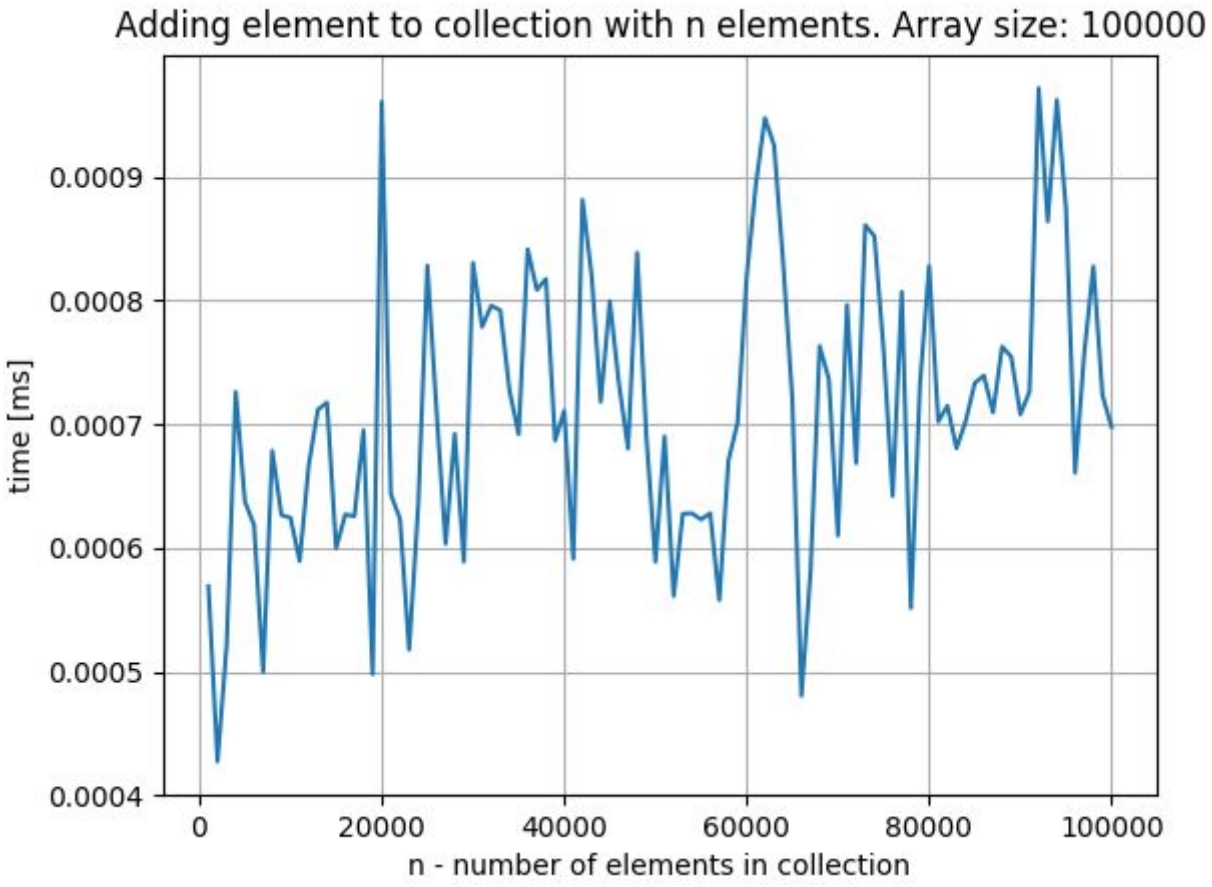
Add

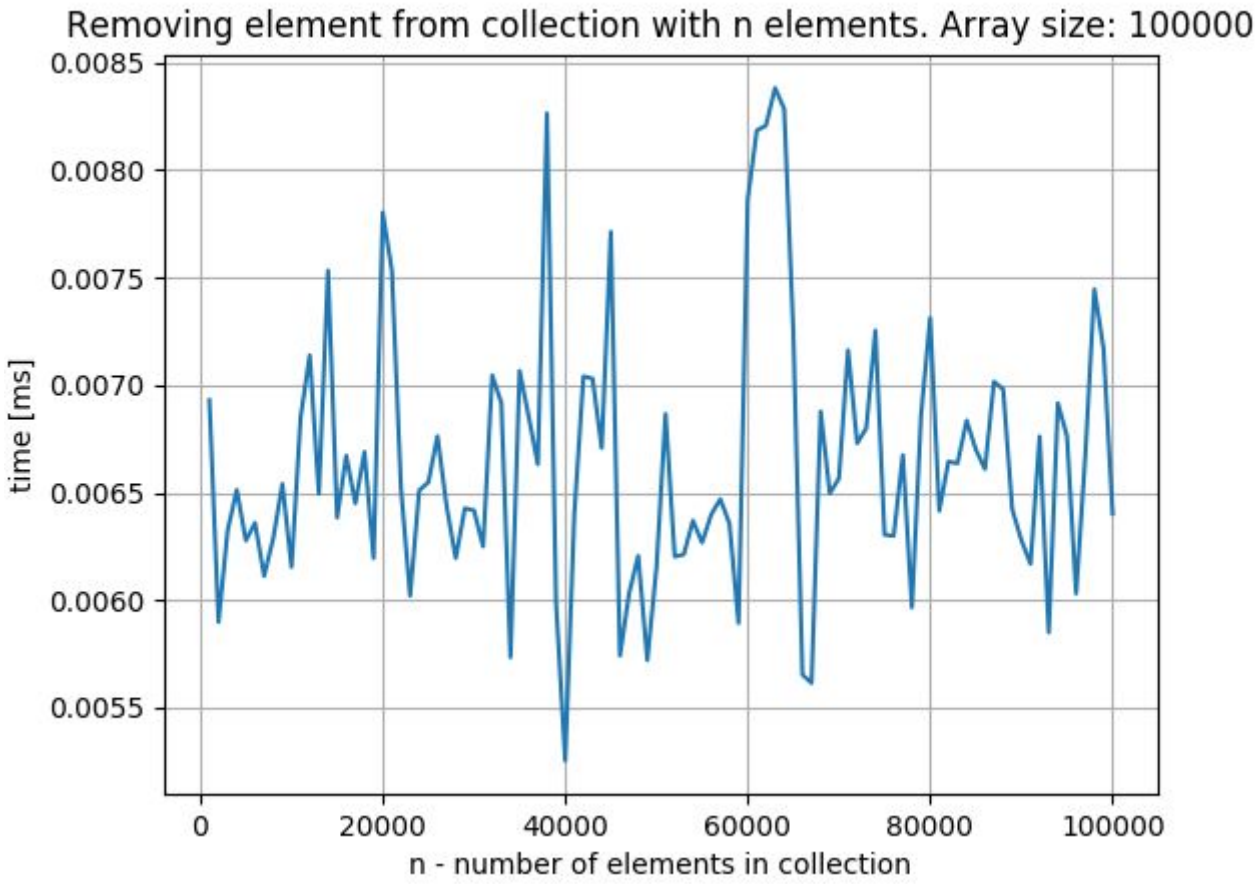
n	t(n)	q(n)
10000	0.0020514	0.119688
20000	0.0033054	0.192852
30000	0.0076302	0.44518
40000	0.0115244	0.672384
50000	0.0277088	1.61665
60000	0.0171396	1
70000	0.0142716	0.832668
80000	0.0269998	1.57529
90000	0.0291168	1.6988
100000	0.0360154	2.1013

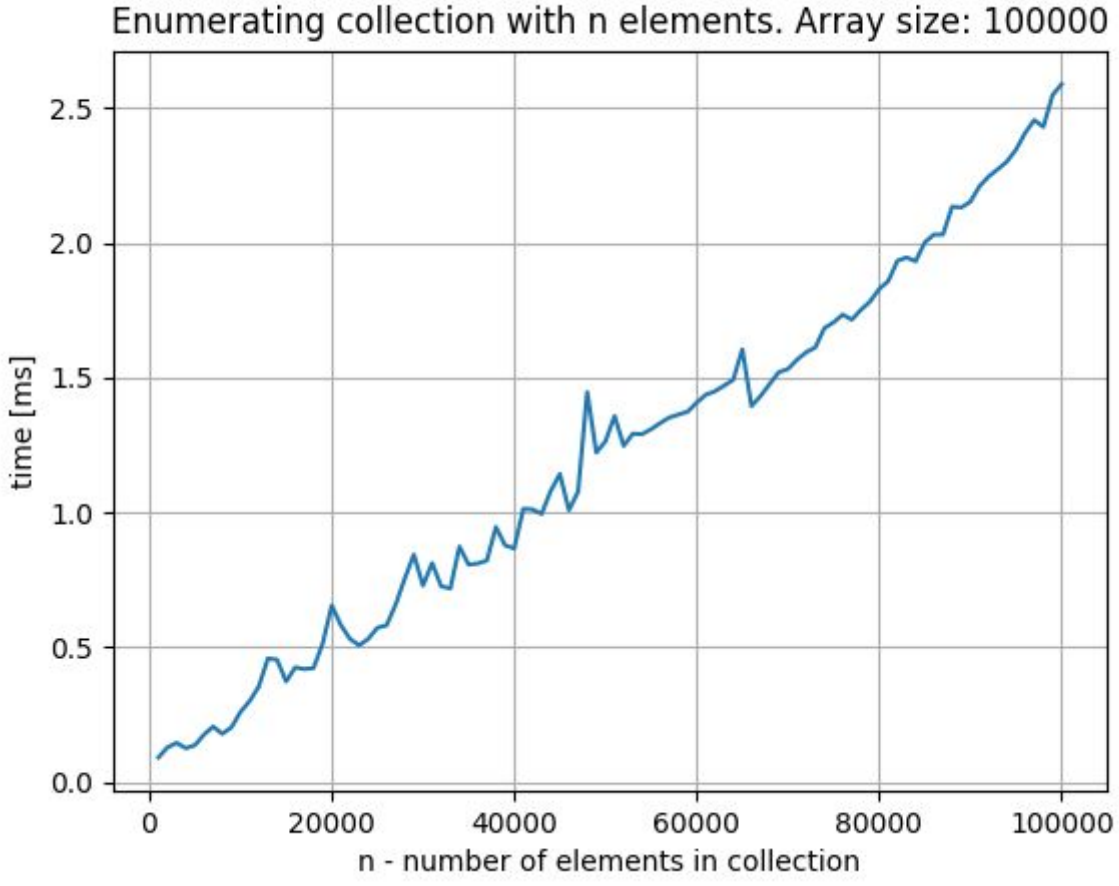
Enumerate

n	t(n)	q(n)
10000	0.146469	0.342319
20000	0.292757	0.342106
30000	0.548143	0.427029
40000	0.987139	0.57677
50000	1.88518	0.881184
60000	2.56724	1
70000	3.86941	1.29191
80000	4.22135	1.23324
90000	5.72278	1.4861
100000	6.26774	1.46486

7.1.2. Rozmiar tablicy: 100 000, maksymalna ilość elementów 100 000




Removing element from collection with n elements. Array size: 100000
time [ms]
0.0085
0.0080
0.0075
0.0070
0.0065
0.0060
0.0055
0
20000
40000
60000
80000
100000
n - number of elements in collection


Enumerating collection with n elements. Array size: 100000
time [ms]
2.5
2.0
1.5
1.0
0.5
0.0
0
20000
40000
60000
80000
100000
n - number of elements in collection

Remove

n	t(n)	q(n)
10000	0.006154	0.782693
20000	0.0078026	0.992369
30000	0.0064172	0.816168
40000	0.0052534	0.66815
50000	0.0061484	0.781981
60000	0.0078626	1
70000	0.0065656	0.835042
80000	0.007312	0.929972
90000	0.0062786	0.79854
100000	0.0064018	0.814209

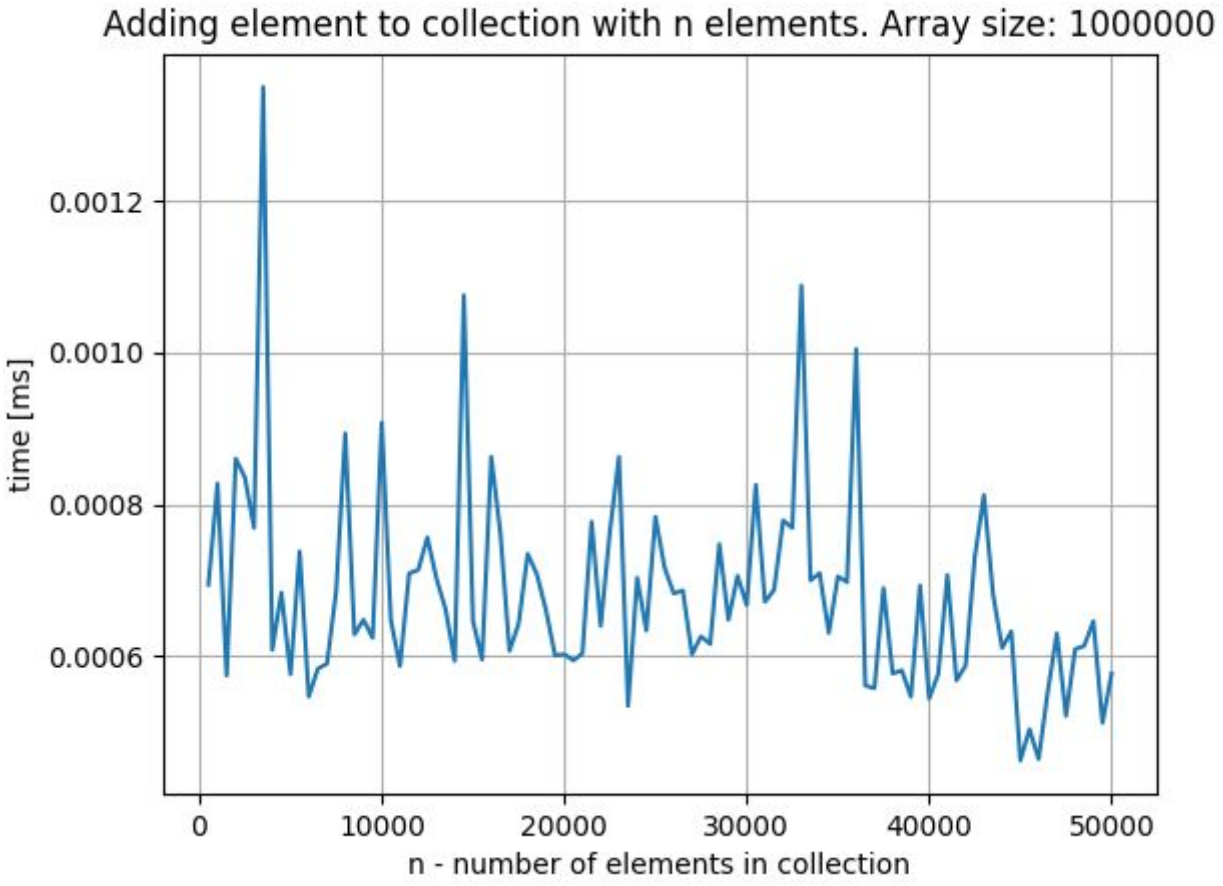
Add

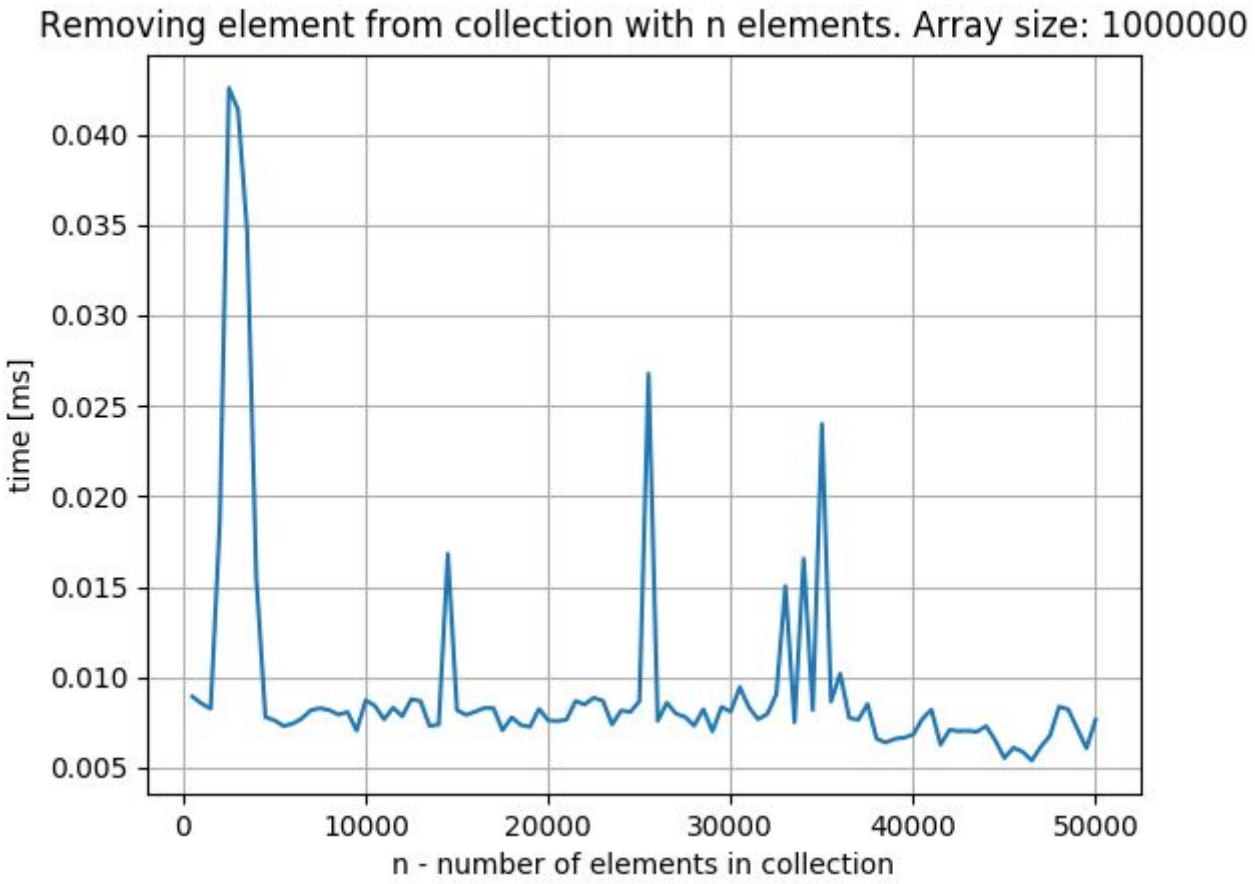
n	t(n)	q(n)
10000	0.0006244	0.762021
20000	0.000961	1.17281
30000	0.0008306	1.01367
40000	0.000711	0.867708
50000	0.0005888	0.718575
60000	0.0008194	1
70000	0.00061	0.744447
80000	0.000828	1.0105
90000	0.000708	0.864047
100000	0.0006978	0.851599

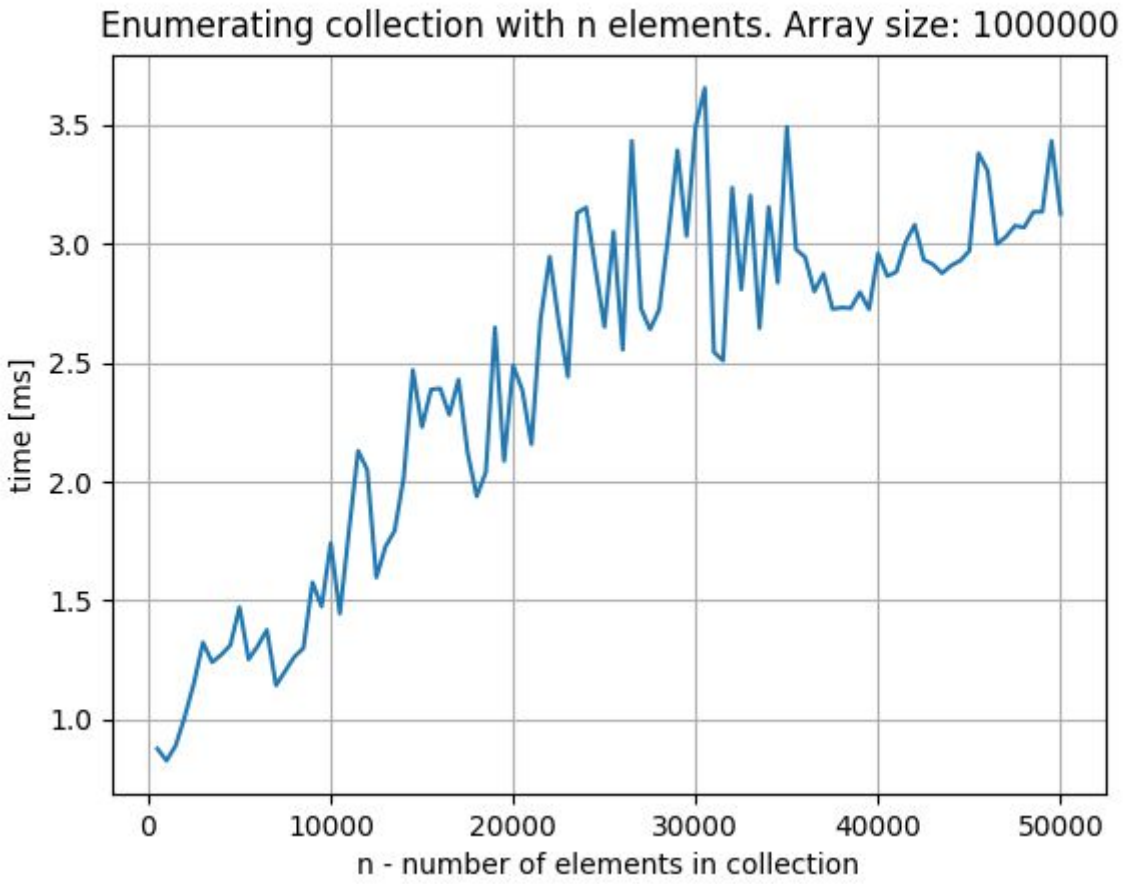
Enumerate

n	t(n)	q(n)
10000	0.260765	1.11202
20000	0.655707	1.39812
30000	0.729201	1.03655
40000	0.866862	0.924174
50000	1.26403	1.07808
60000	1.40698	1
70000	1.53177	0.933166
80000	1.82845	0.974671
90000	2.15231	1.01983
100000	2.58815	1.1037

7.1.3. Rozmiar tablicy: 1 000 000, maksymalna ilość elementów 50 000




Removing element from collection with n elements. Array size: 1000000
time [ms]
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
10000
20000
30000
40000
50000
n - number of elements in collection



Remove

n	t(n)	q(n)
5000	0.0076208	0.940886
10000	0.0087328	1.07818
15000	0.0081816	1.01012
20000	0.0076158	0.940269
25000	0.0086808	1.07176
30000	0.0080996	1
35000	0.0239946	2.96244
40000	0.0068338	0.843721
45000	0.005539	0.683861
50000	0.0076564	0.945281

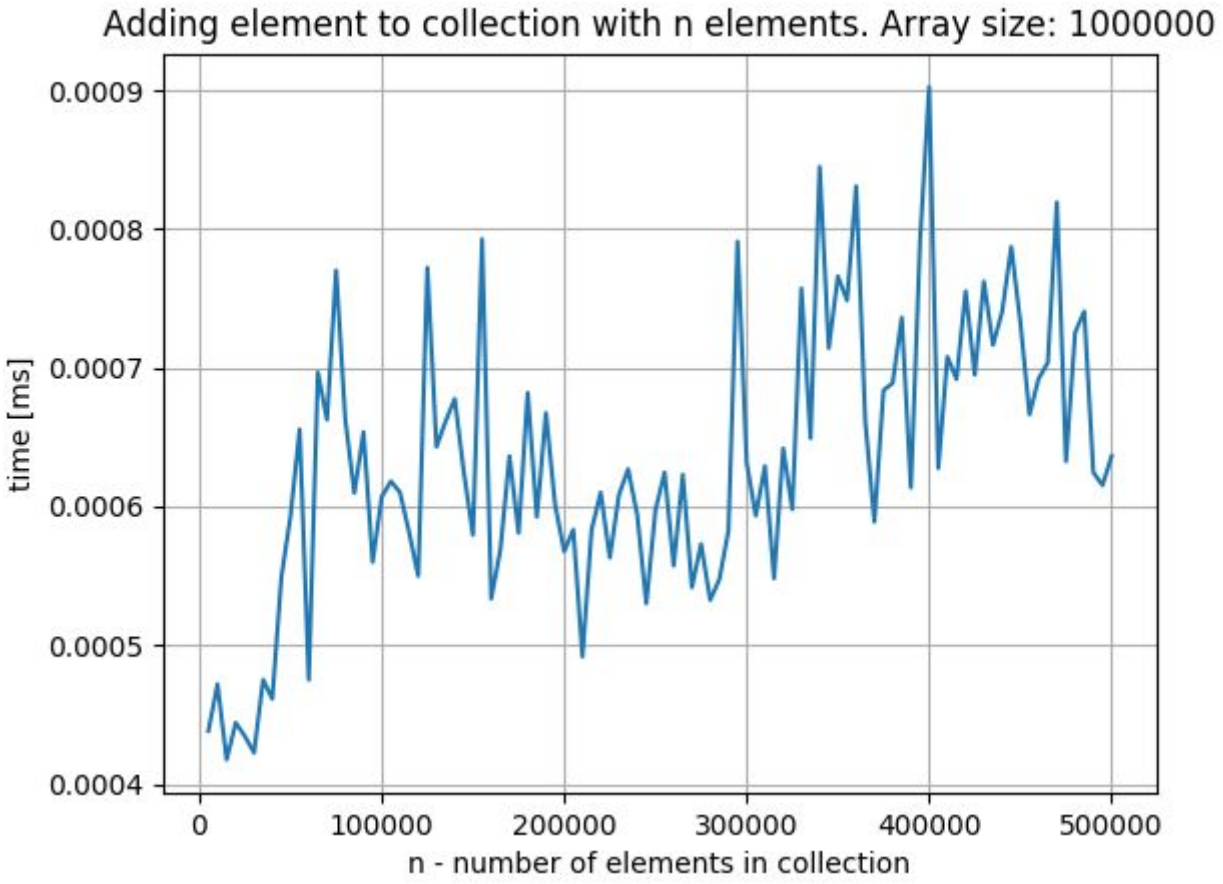
Add

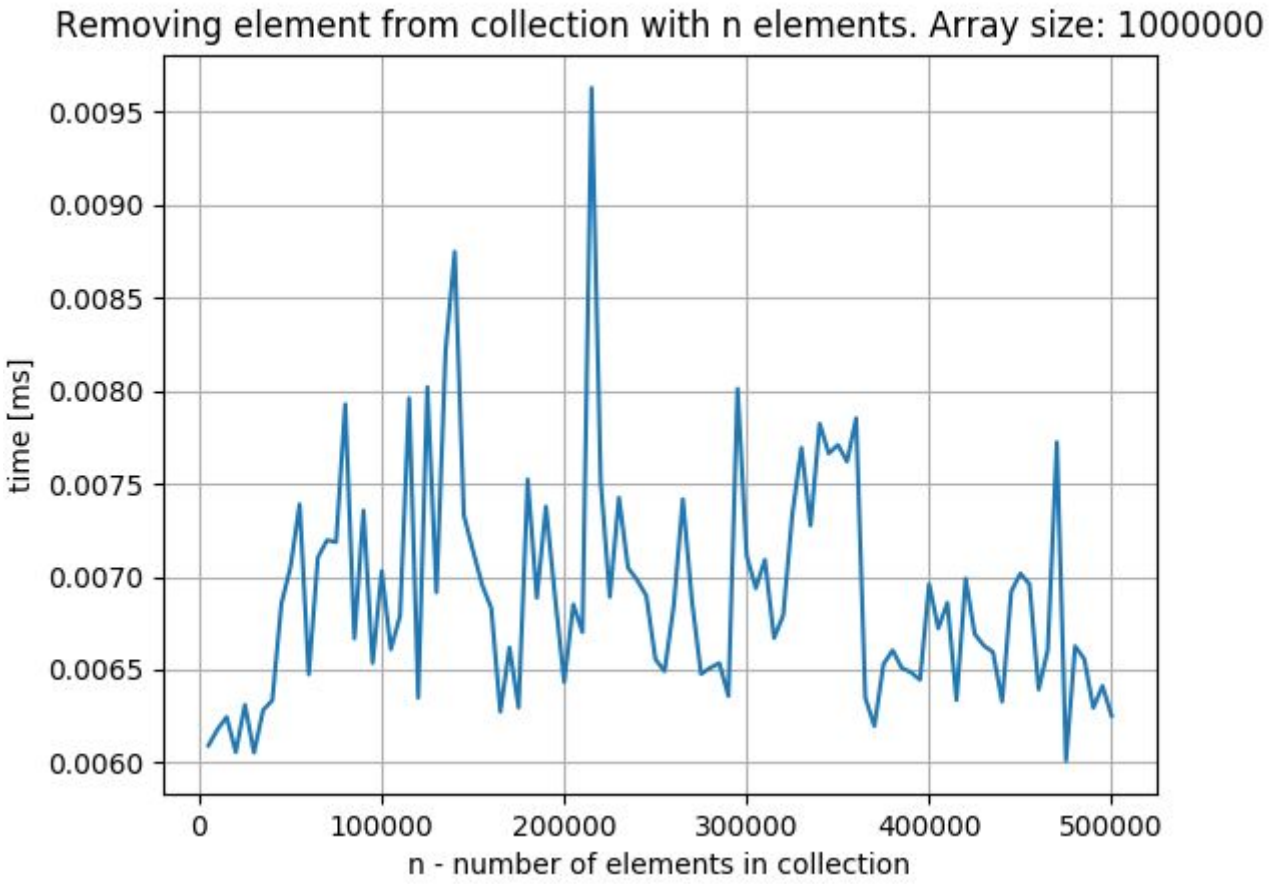
n	t(n)	q(n)
5000	0.0005764	0.863391
10000	0.000908	1.3601
15000	0.0006466	0.968544
20000	0.0006028	0.902936
25000	0.000784	1.17436
30000	0.0006676	1
35000	0.0007054	1.05662
40000	0.0005444	0.815458
45000	0.0004628	0.693229
50000	0.0005774	0.864889

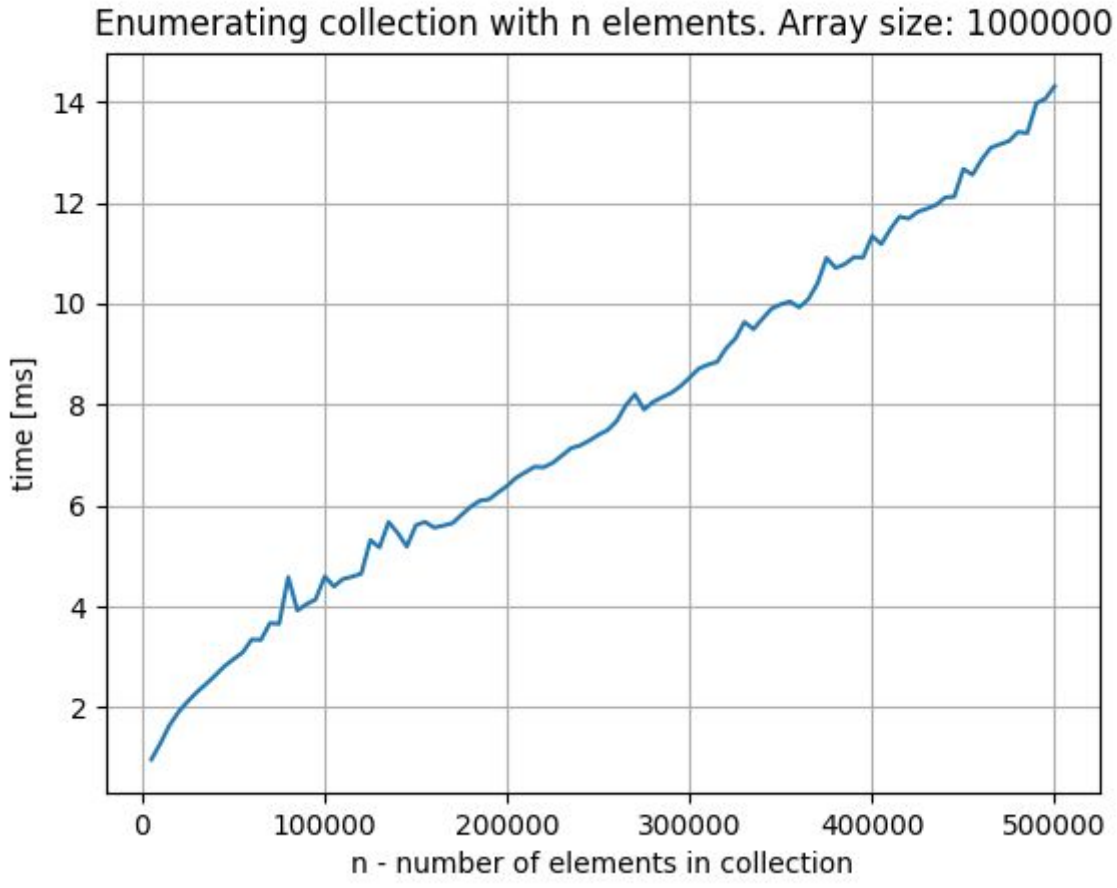
Enumerate

n	t(n)	q(n)
5000	1.47258	2.52851
10000	1.74177	1.49536
15000	2.23028	1.27651
20000	2.48861	1.06828
25000	2.6512	0.910457
30000	3.49434	1
35000	3.49237	0.856662
40000	2.95985	0.635282
45000	2.97103	0.566828
50000	3.12608	0.536769

7.1.4. Rozmiar tablicy: 1 000 000, maksymalna ilość elementów 500 000




Removing element from collection with n elements. Array size: 1000000
time [ms]
0.0095
0.0090
0.0085
0.0080
0.0075
0.0070
0.0065
0.0060
0
100000
200000
300000
400000
500000
n - number of elements in collection


Enumerating collection with n elements. Array size: 1000000
time [ms]
14
12
10
8
6
4
2
0
100000
200000
300000
400000
500000
n - number of elements in collection

Remove

n	t(n)	q(n)
50000	0.0070526	0.991481
100000	0.0070302	0.988332
150000	0.0071408	1.00388
200000	0.006434	0.904516
250000	0.0065562	0.921695
300000	0.0071132	1
350000	0.0077096	1.08384
400000	0.0069604	0.978519
450000	0.0070176	0.98656
500000	0.0062516	0.878873

Add

n	t(n)	q(n)
50000	0.0005932	0.939202
100000	0.000607	0.961051
150000	0.0005796	0.917669
200000	0.0005676	0.89867
250000	0.0005974	0.945852
300000	0.0006316	1
350000	0.000766	1.21279
400000	0.0009024	1.42875
450000	0.0007326	1.15991
500000	0.0006364	1.0076

Enumerate

n	t(n)	q(n)
50000	2.95829	2.0798
100000	4.59569	1.61548
150000	5.61368	1.31555
200000	6.39147	1.12337
250000	7.40152	1.04071
300000	8.53436	1
350000	9.98727	1.00306
400000	11.3392	0.99649
450000	12.6664	0.989445
500000	14.308	1.00591

7.2. Wnioski

Interpretując wyniki trzeba mieć na uwadze, że mimo uśrednienia 5 pomiarów dla każdej kombinacji parametrów, występuje czynnik losowy wynikający z metody pomiaru. Na podstawie wyników można zauważyć, zgodnie z oczekiwaniami, że czas dodawania i usuwania jest w przybliżeniu stały, a czas enumeracji rośnie liniowo. Jest to prawdziwe jednak tylko przy dostatecznie dużym rozmiarze tablicy wewnętrznej. Przy małej wielkości tablicy, a dużej ilości elementów, występują kolizje i czas dodawania oraz usuwania elementów rośnie liniowo. Dla przypadków kiedy użyta tablica jest wystarczająco duża, zgodność teoretycznego czasu działania z otrzymanym jest zadowalająca.