Wrocław, 02.03.18

Prowadzący: dr inż. Zbigniew Buchalski

Filip Mazur 226018

Struktury Danych i Złożoność Obliczeniowa

Sprawozdanie - Projekt nr 1

Temat: *Badanie Efektywności poszczególnych operacji na różnych strukturach.*

1. **Używane struktury oraz operacje na nich:**

   a) **Tablica**

      i. Dodawanie i Usuwanie

         - Na początek
           Dodawanie polega na przesunięciu każdego elementu na następne miejsce oraz dodaniu nowego na początek.
           Usunięcie z początku zaczyna się od usunięcia pierwszego elementu, po czym pozostałe w tablicy zostają przesunięte o jedno miejsce wstecz.
           Złożoność czasowa takich operacji zależy wprost proporcjonalnie od ilości elementów - O(n).
         - Na koniec
           Złożoność w przypadku usunięcia jak i dodania elementu na koniec wynosi O(1) [czas jest stały]. Wystąpić może także dodatkowy czas w przypadku relokacji pamięci. Jej złożoność zależy od ilości elementów w tablicy
         - Dowolne miejsce w tablicy
           Czas, w którym operacja zostanie wykonana zależy od położenia modyfikowanego elementu w tablicy. Pesymistyczne założenie przewiduje złożoność O(n).

      ii. Wyszukiwanie
      Jeśli szukany element znajduje się na końcu tablicy, to jego złożoność czasowa wynosi O(n) [Najgorszy wypadek].

   b) **Lista Dwukierunkowa**

      i. Dodawanie i usuwanie

         - Początek i koniec listy
           Czas zarówno usunięcia jak i dodania elementu na początek będzie stały. Złożoność tej operacji to O(n).
         - Dowolne miejsce na liście
           Czas przeprowadzenia tej operacji jest zależny od położenia modyfikowanego elementu. Znalezienie go wymaga przejścia listy do poprzednika danego elementu, a następnie dokonać jego usunięcia lub dodania. W najgorszym przypadku jego złożoność to O(n).

      ii. Wyszukiwanie
      Wyszukując element o podanym kluczu przechodzimy listę aż do poprzednika tego elementu. W najgorszym wypadku operacja będzie miała złożoność obliczeniową O(n).

c) Kopiec Binarny

i. Dodawanie i usuwanie(elementu o podanym kluczu)
Zarówno dodanie jak i usunięcie elementu może naruszyć warunek kopca, co będzie wymagało odtworzenia jego struktury. Pesymistyczny przypadek zakłada złożoność zależną od liczby poziomów drzewa wynoszącą $O(n*\log_2 n)$. Nie naruszenie warunku sprawia, że złożoność czasowa jest stała [czyli O(1)]. Element zostaje dodany do tablicy reprezentującej kopiec na ostatnim miejscu jako ostatni liść.

ii. Wyszukiwanie
Wyszukiwanie w kopcu odbywa się tak samo jak w tablicy. Złożoność obliczeniowa w tym przypadku wynosi najwyżej O(n).

d) Drzewo BST

i. Dodawanie elementu(o podanym kluczu)
Dodawanie elementu w drzewie BST polega na doczepieniu nowego elementu jako liścia ostatniego

ii. Usuwanie Elementu
Usuwanie elementu z drzewa BST uwzględnia 3 przypadki. Pierwszy z nich to, usuwany element nie posiadający synów. Wtedy po prostu go odłączamy od drzewa. Drugim oraz trzecim są przypadki gdy, 1. Usuwany węzeł posiada tylko lewego potomka, wtedy usuwając go zastępujemy go potomkiem. 2. Węzeł posiada obu potomków. W tym wypadku usuwany węzeł zastępujemy losowo następnikiem albo poprzednikiem.

iii. Wyszukiwanie
Przeszukiwanie Drzewa BST rozpoczynamy od korzenia. Sprawdzane są warunki czy przeszukiwany element jest większy czy mniejszy od korzenia. Jeżeli jest większy to przechodzimy do prawego poddrzewa, jeżeli jest mniejszy to przechodzimy do przeszukiwania lewego poddrzewa. Elementy najmniejszy i największy znajdują się zawsze najbardziej odpowiednio po lewej i po prawej.

2. Plan eksperymentu

a) Zarówno tablica jak i tablica reprezentująca kopiec są dynamicznie relokowane przy usuwaniu jak i dodawaniu elementów. Czas relokacji w niektórych przypadkach nie był brany pod uwagę aby uzyskać dokładniejsze wyniki.
b) Każda operacja jest testowana na strukturach zawierających 1000, 2000, 5000, 10000 oraz 20000 elementów. Dla większości pomiarów wygenerowana zostanie nowa populacja. Wyniki pomiarów zostaną uśrednione i umieszczone w tabeli a potem przedstawione na wykresie.
c) Elementem struktur będzie wygenerowana losowo z przedziału <-1000, 1000> liczba całkowita typu integer. Generacja zostanie wykonana za pomocą funkcji rand().
d) Do pomiarów czasu wykorzystano funkcję:
QueryPerformanceCounter(_otut LARGE_INTEGER *lpPerformanceCount);

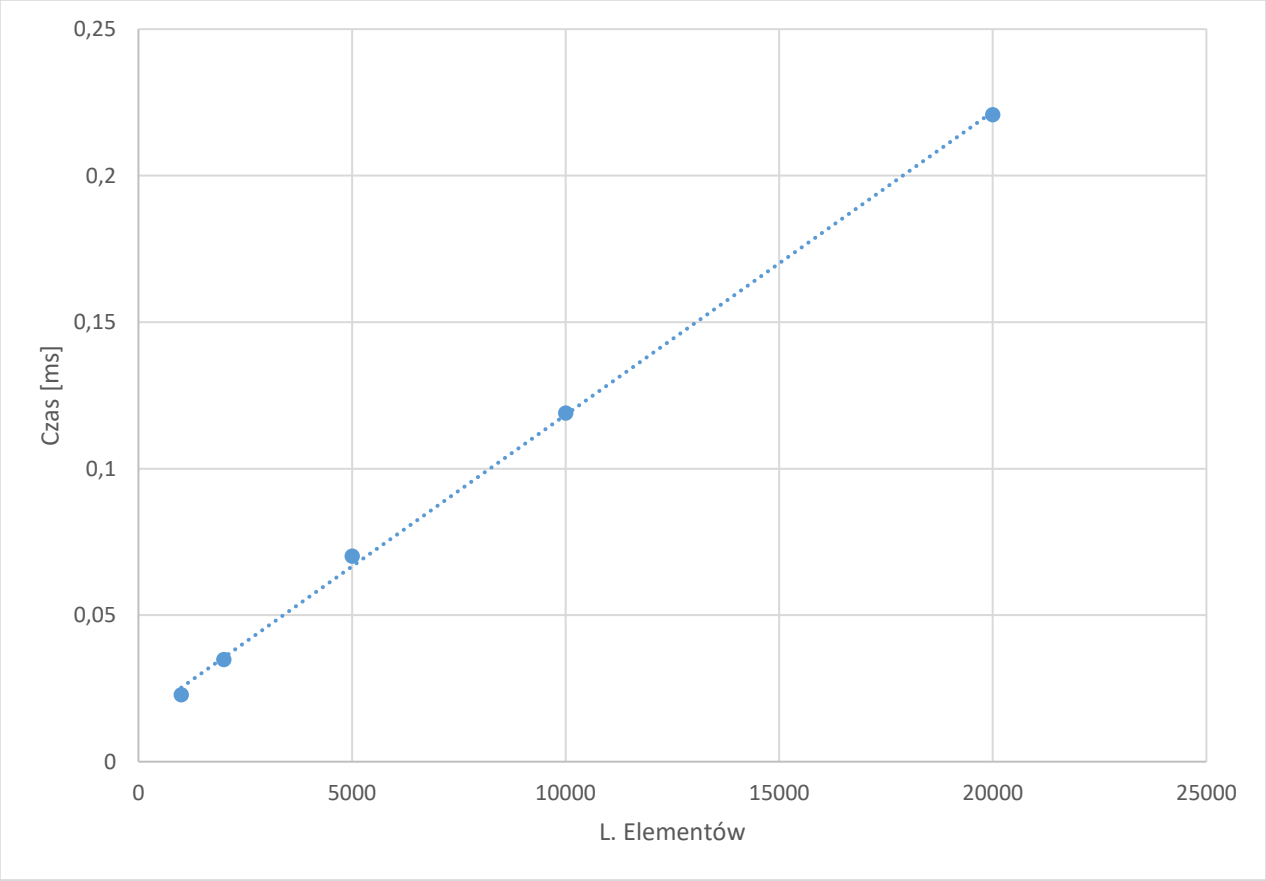
3. Wyniki eksperymentów:

a) Tablica:

i. Usuwanie z początku tablicy

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,02284
2	2000	0,03486
3	5000	0,07012
4	10000	0,119
5	20000	0,2208

Tabela 1 – Usuwanie elementu z początku tablicy

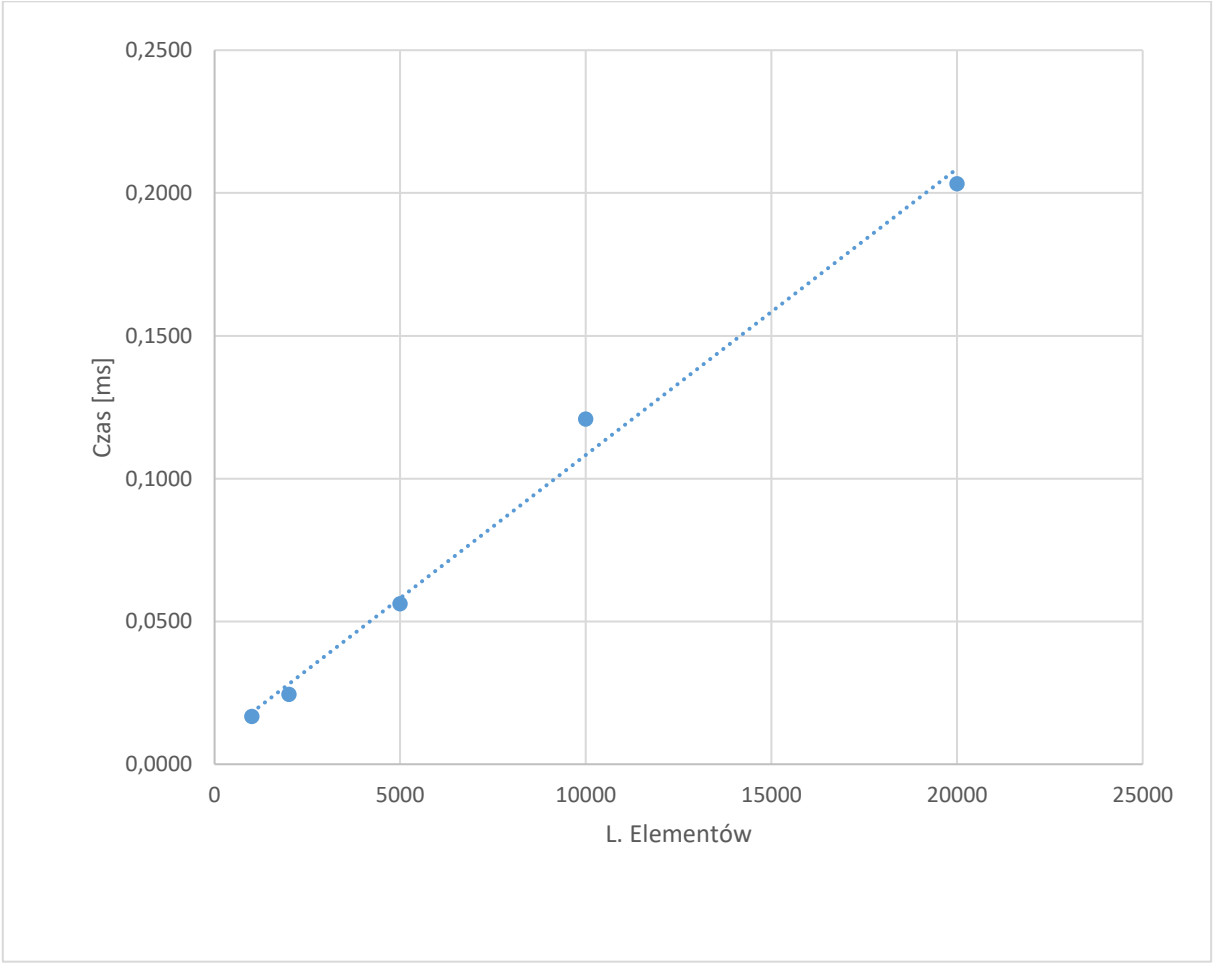


Wykres 1 – Zależność czasu potrzebnego na usunięcie od ilości elementów

ii. Dodawanie na początek tablicy

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,0166
2	2000	0,0244
3	5000	0,0561
4	10000	0,1208
5	20000	0,2032

Tabela 2 – Dodawanie elementu na początek tablicy

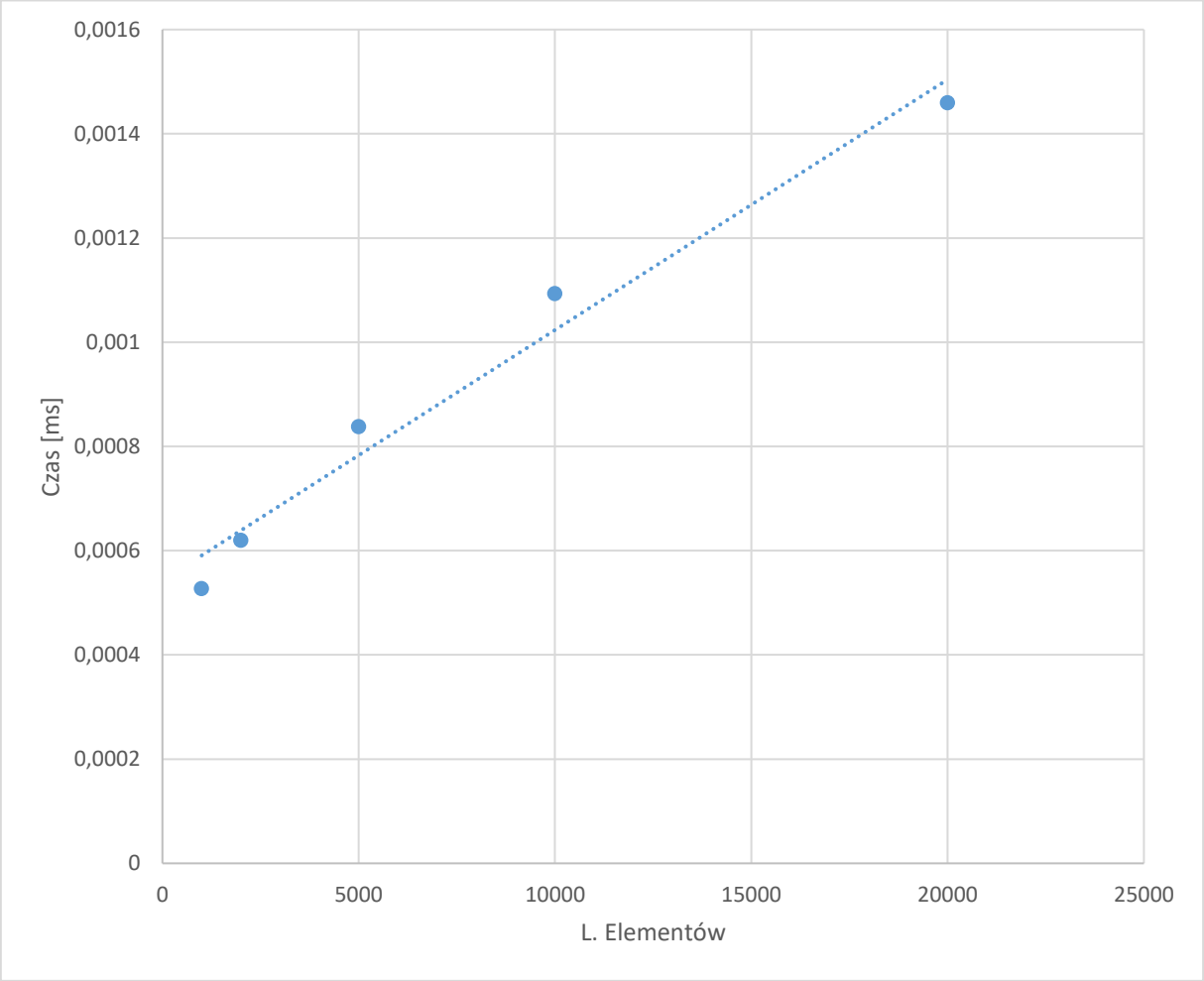


Wykres 2 – Zależność czasu potrzebnego na dodanie od ilości elementów

iii. Usuwanie z końca tablicy

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,0005278
2	2000	0,00062
3	5000	0,0008386
4	10000	0,001094
5	20000	0,00146

Tabela 3 – Usuwanie z końca tablicy

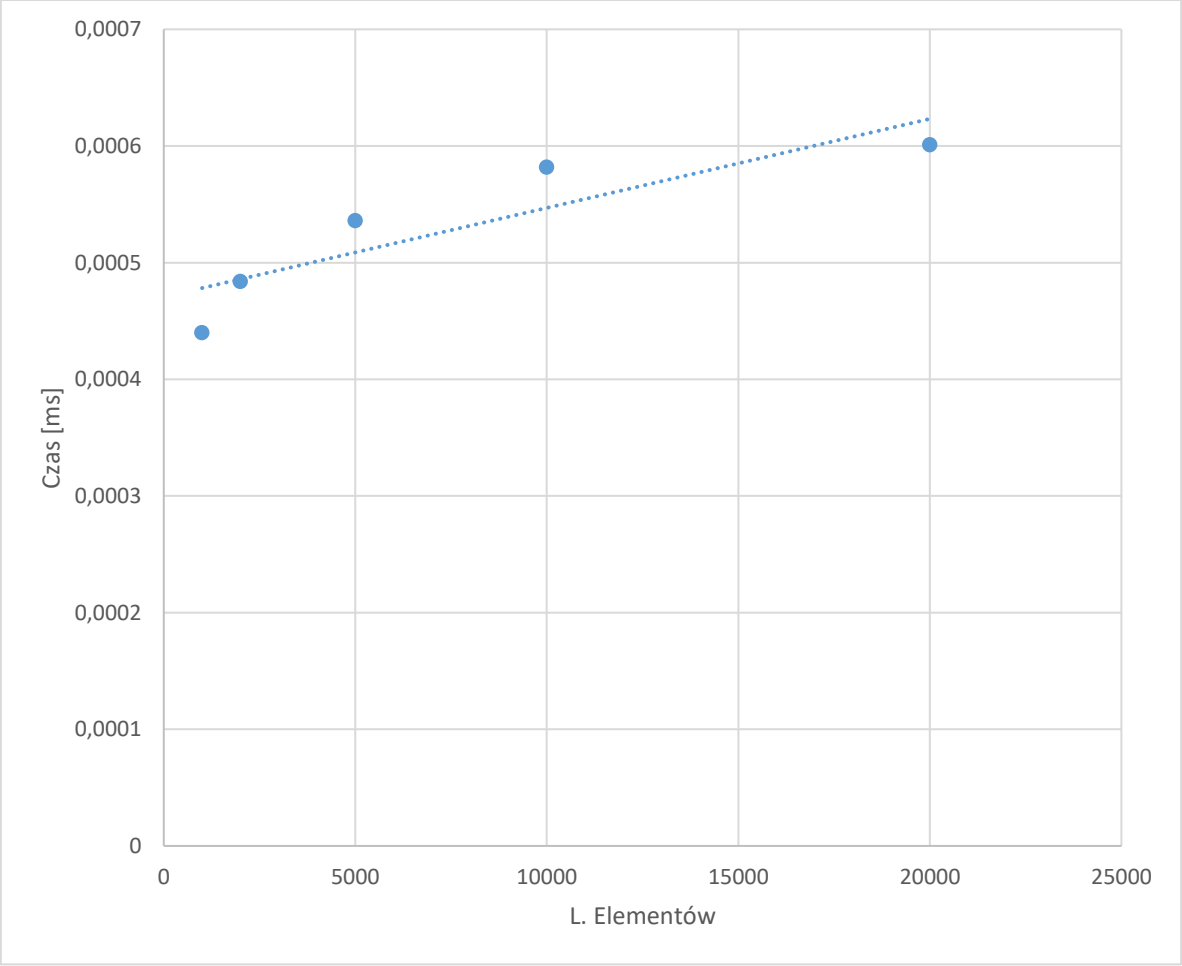


Wykres 3 – Zależność czasu potrzebnego do usunięcia z końca od ilości elementów

iv. Dodawanie na koniec tablicy

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,00044
2	2000	0,000484
3	5000	0,000536
4	10000	0,000582
5	20000	0,000601

Tabela 4 – Dodawanie elementu na koniec tablicy

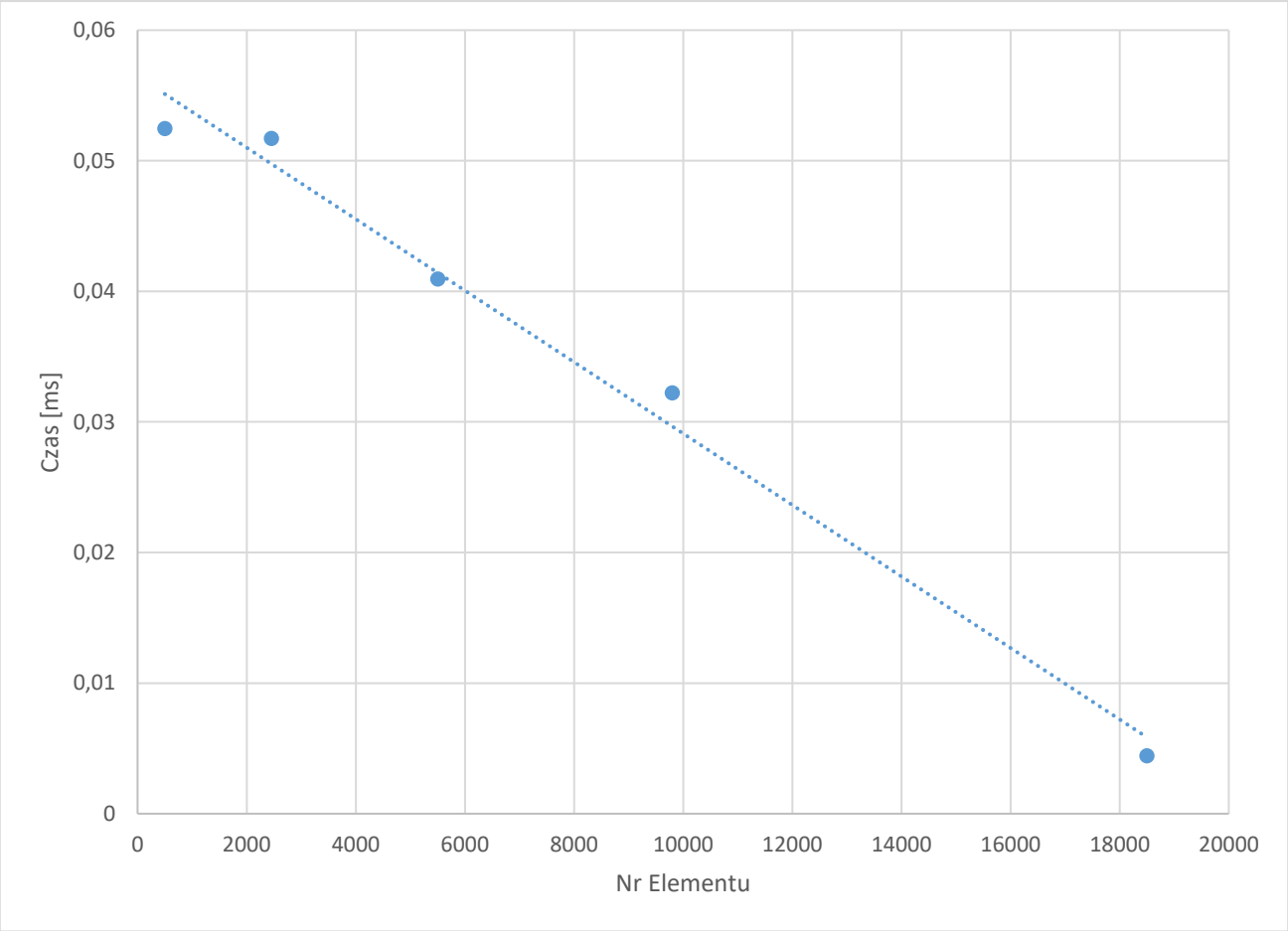


Wykres 4 – Zależność czasu potrzebnego na dodanie od Ilości elementów

v. Usuwanie z wnętrza tablicy

Lp.	Numer indeksu	Średni czas [ms]
1	500	0,05248
2	2450	0,05172
3	5500	0,04094
4	9800	0,03224
5	18500	0,00444

Tabela 5 – Usuwanie z wybranych miejsc wewnątrz tablicy

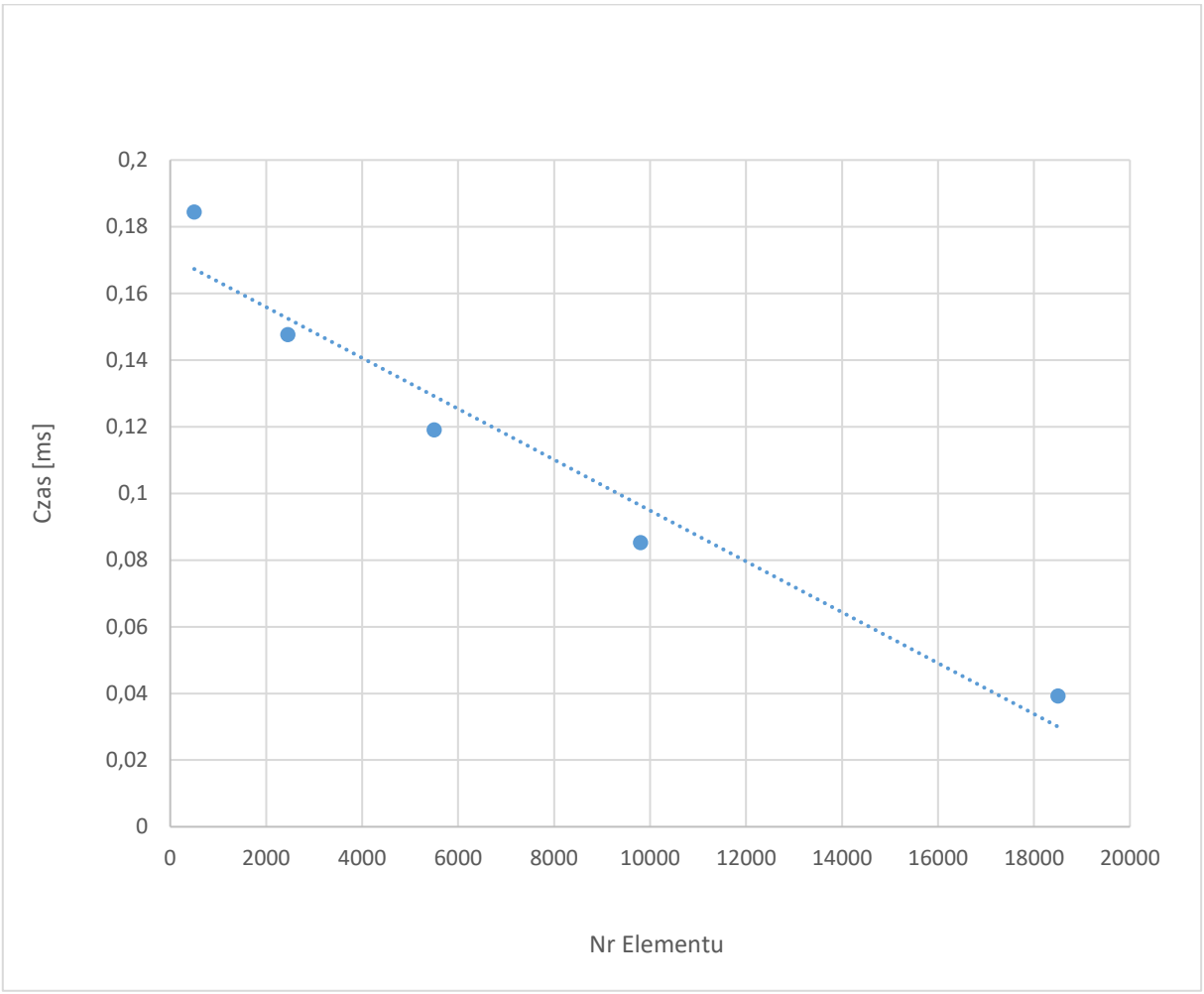


Wykres 5 – Zależność czasu potrzebnego na usunięcie od indeksu zadanego elementu.

vi. Dodawanie do wnętrza tablicy

Lp.	Numer indeksu	Średni czas [ms]
1	500	0,1844
2	2450	0,1476
3	5500	0,119
4	9800	0,0852
5	18500	0,03916

Tabela 6 – Dodawanie na wybrane miejsca w tablicy

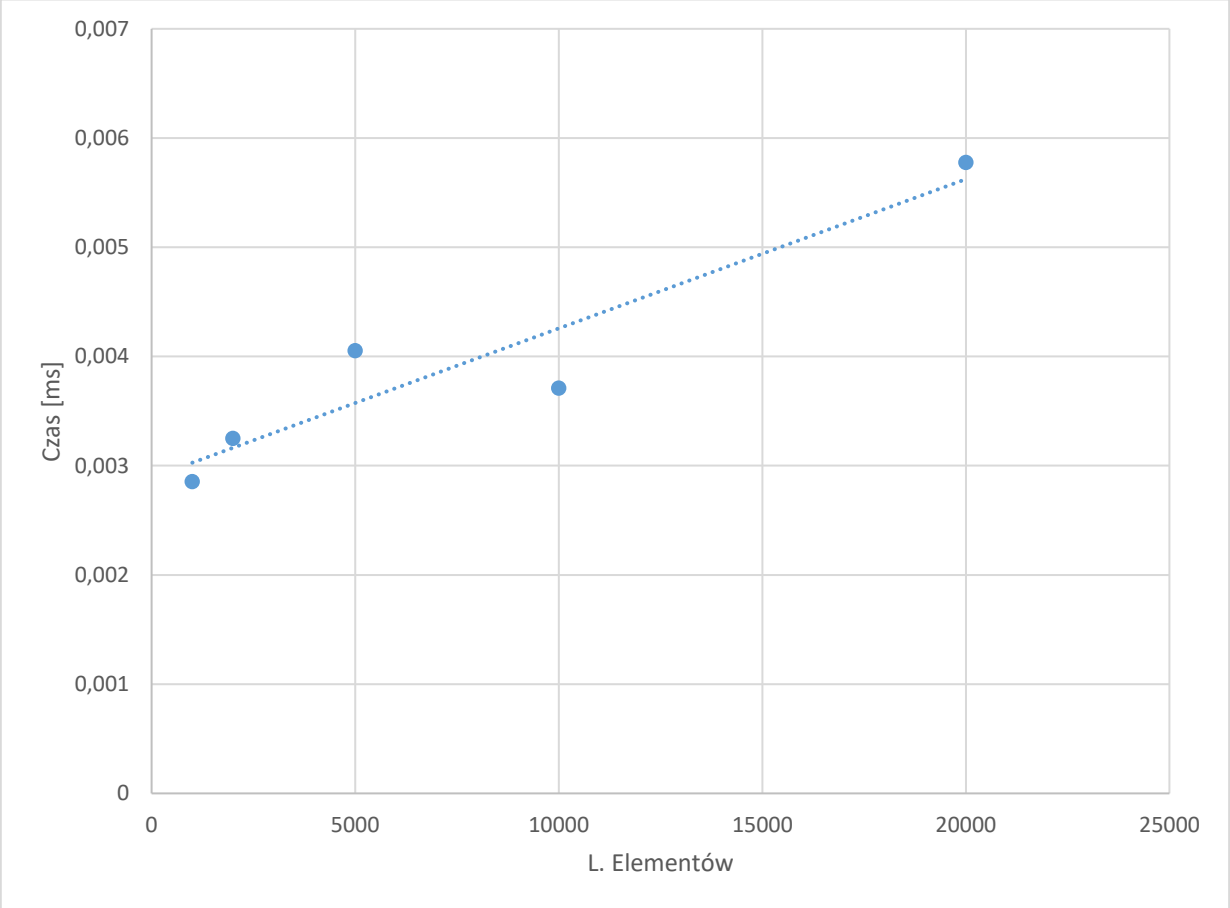


Wykres 6 – zależność czasu potrzebnego na dodanie od indeksu zadanego elementu.

vii. Wyszukiwanie

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,002854
2	2000	0,00325
3	5000	0,004054
4	10000	0,003712
5	20000	0,005776

Tabela 7 – Wyszukiwanie zadanego elementu w tablicy



Wykres 7 – Zależność czasu potrzebnego na wyszukanie zadanego elementu od ilości elementów w tablicy

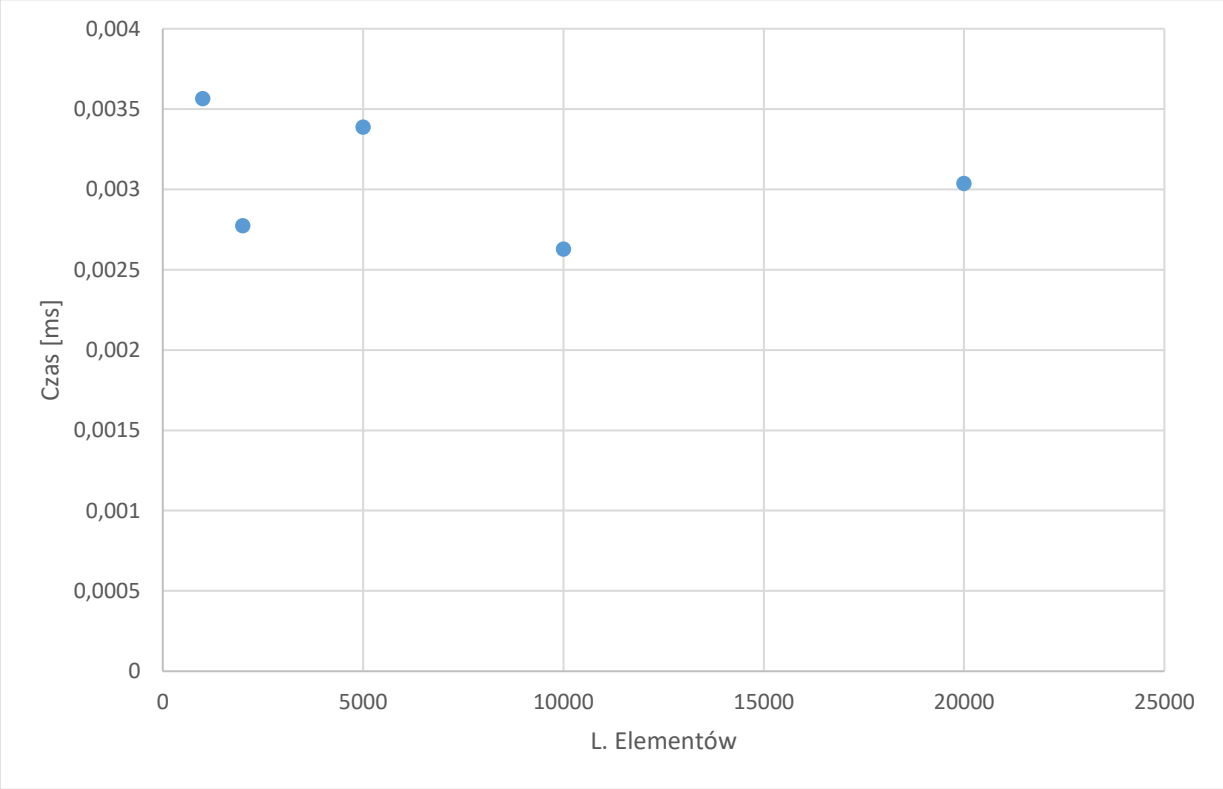
W testach zadbano, aby wszystkie elementy zostały odnalezione w strukturze, przyjmując optymistyczne założenie złożoności obliczeniowej.

b) Lista dwukierunkowa:

i. Dodawanie na początek

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,003566
2	2000	0,002775
3	5000	0,003388
4	10000	0,00263
5	20000	0,003038

Tabela 8 – Dodawanie na początek listy

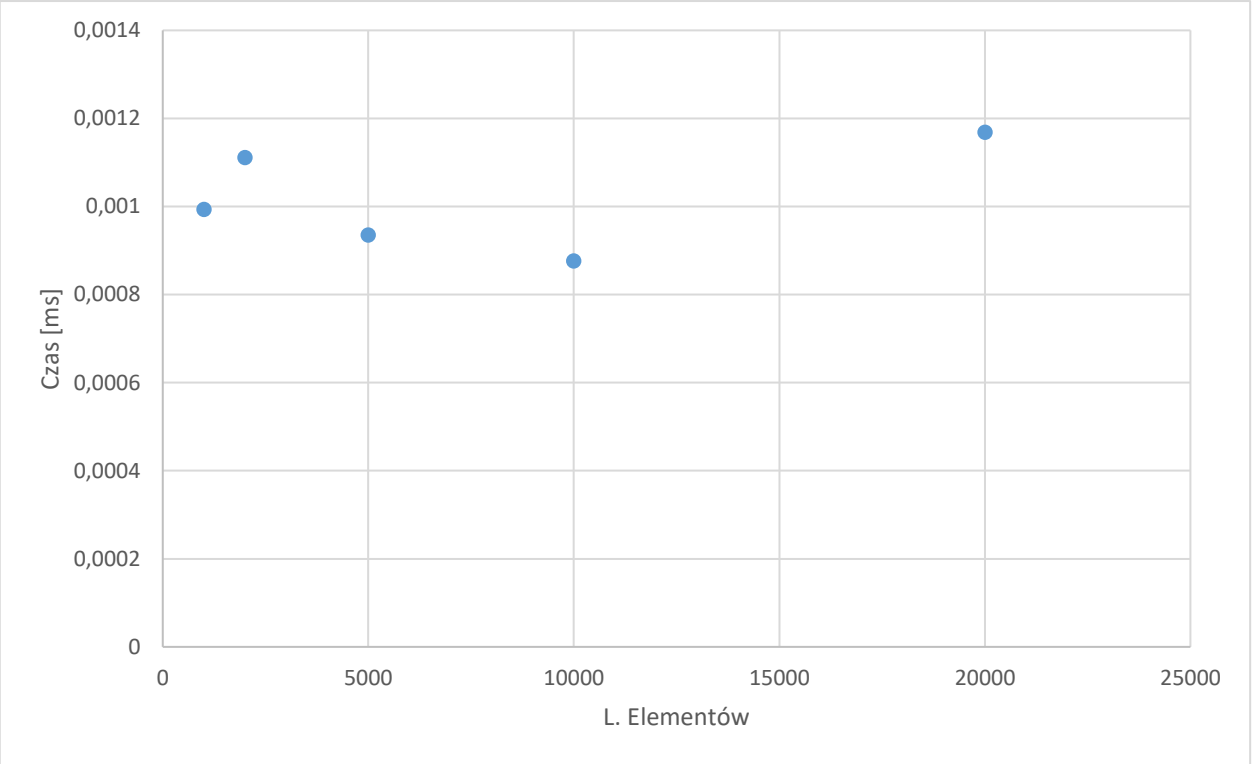


Wykres 8 – Zależność czasu potrzebnego na dodanie na początek od liczby elementów na liście

ii. Usuwanie z początku

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,0009938
2	2000	0,001111
3	5000	0,0009354
4	10000	0,0008768
5	20000	0,0011688

Tabela 9 – Usuwanie elementu z początku listy

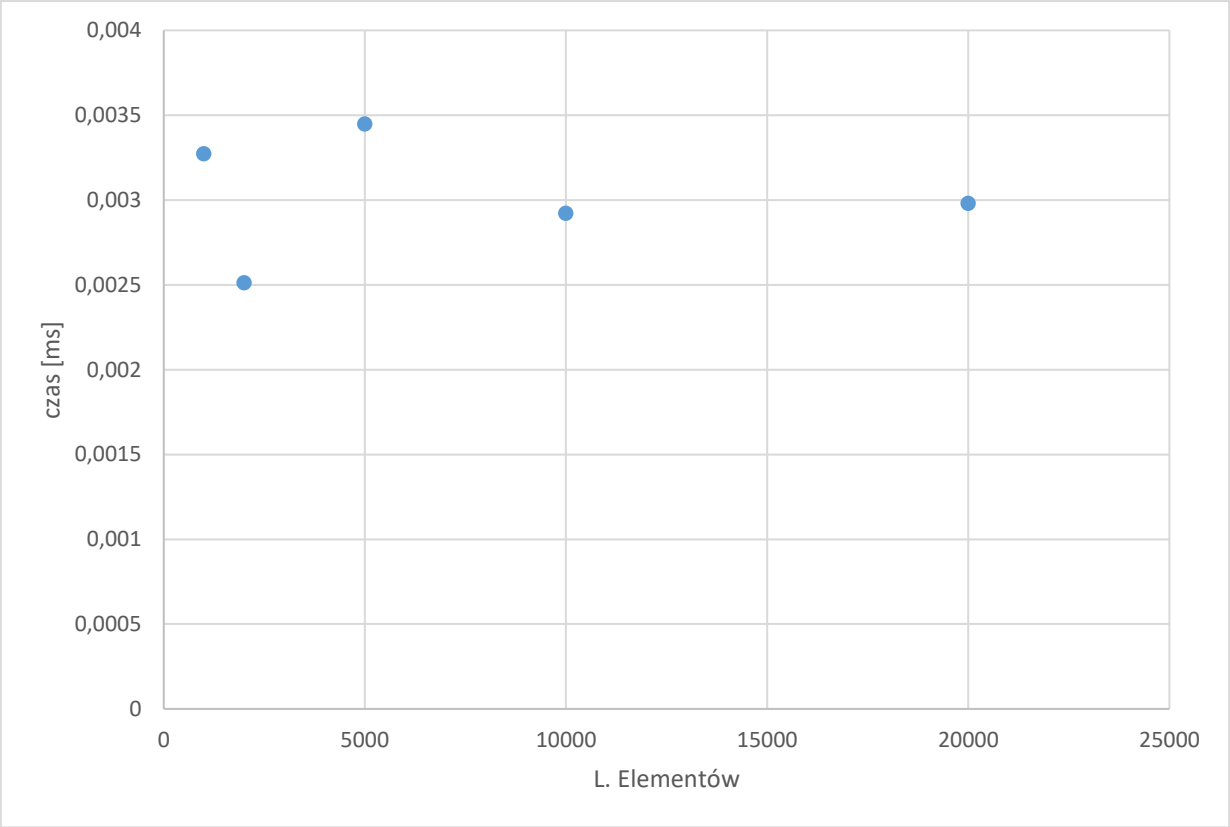


Wykres 9 – Zależność czasu na potrzebnego na usunięcie z początku od ilości elementów na liście

iii. Dodawanie na koniec

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,003274
2	2000	0,002512
3	5000	0,003448
4	10000	0,002922
5	20000	0,00298

Tabela 10 – Dodawanie elementu na koniec listy

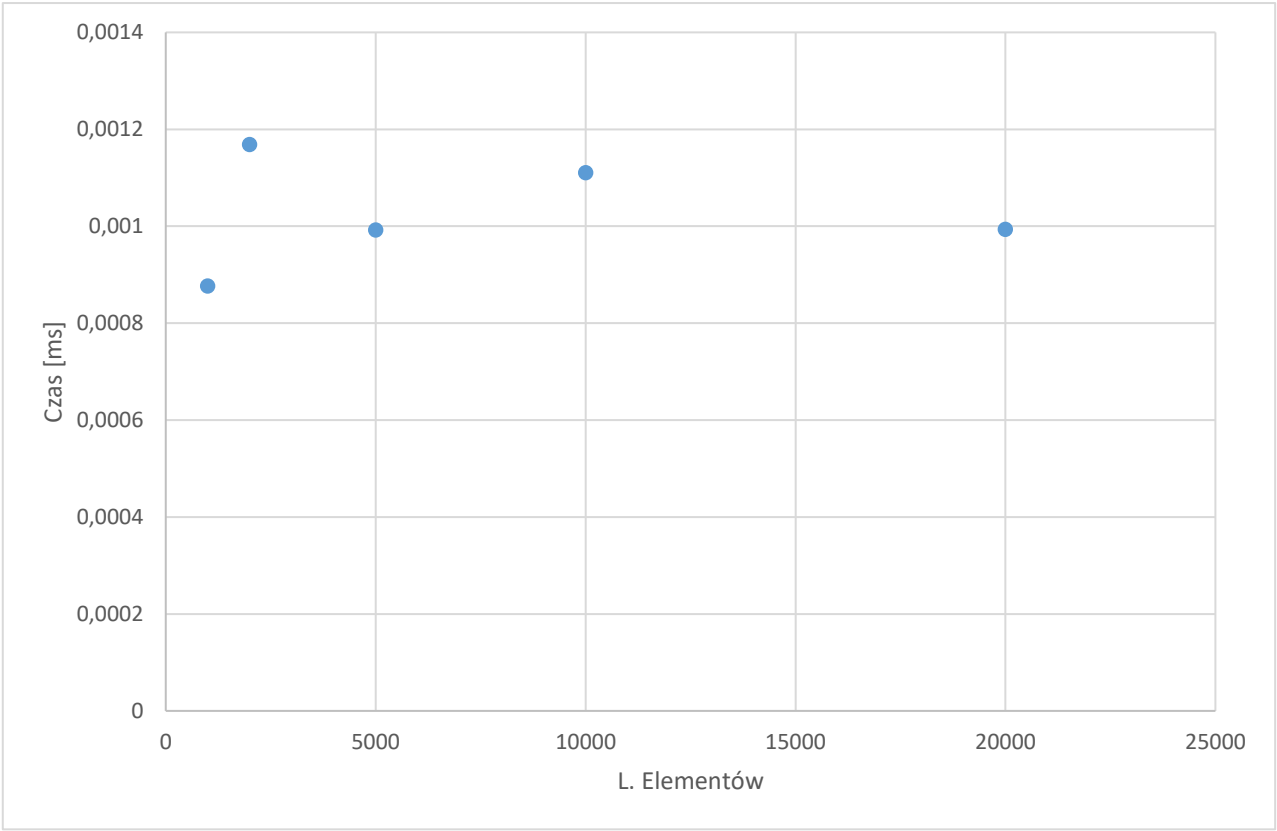


Wykres 10 – Zależność czasu potrzebnego na dodanie na koniec od ilości elementów na liście

iv. Usuwanie z końca

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,0008768
2	2000	0,00116894
3	5000	0,0009926
4	10000	0,0011104
5	20000	0,0009934

Tabela 11 – Usuwanie z końca

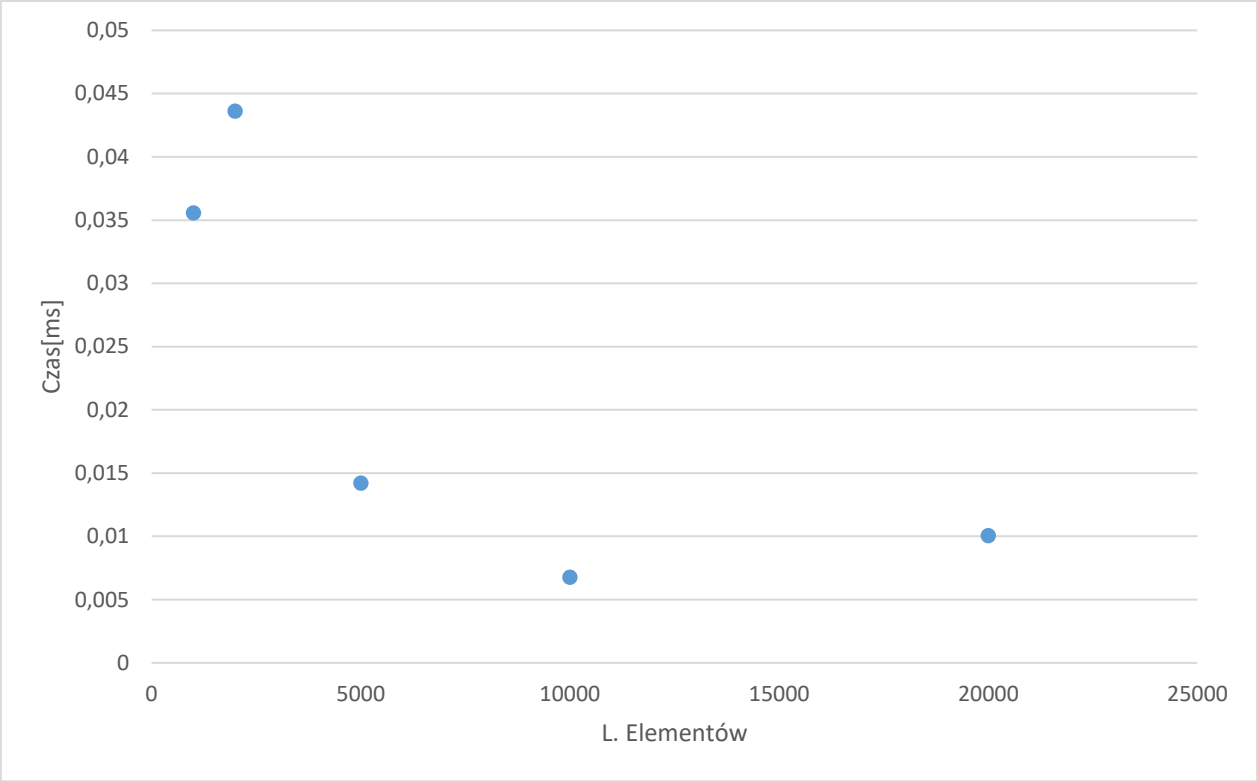


Wykres 11 – Zależność czasu potrzebnego na usunięcie z końca od ilości elementów

v. Dodawanie po indeksie

Lp.	Klucz elementu	Średni czas [ms]
1	1000	0,03558
2	2000	0,04362
3	5000	0,01422
4	10000	0,006782
5	20000	0,010054

Tabela 12 – Dodawanie za wybranym elementem na liście

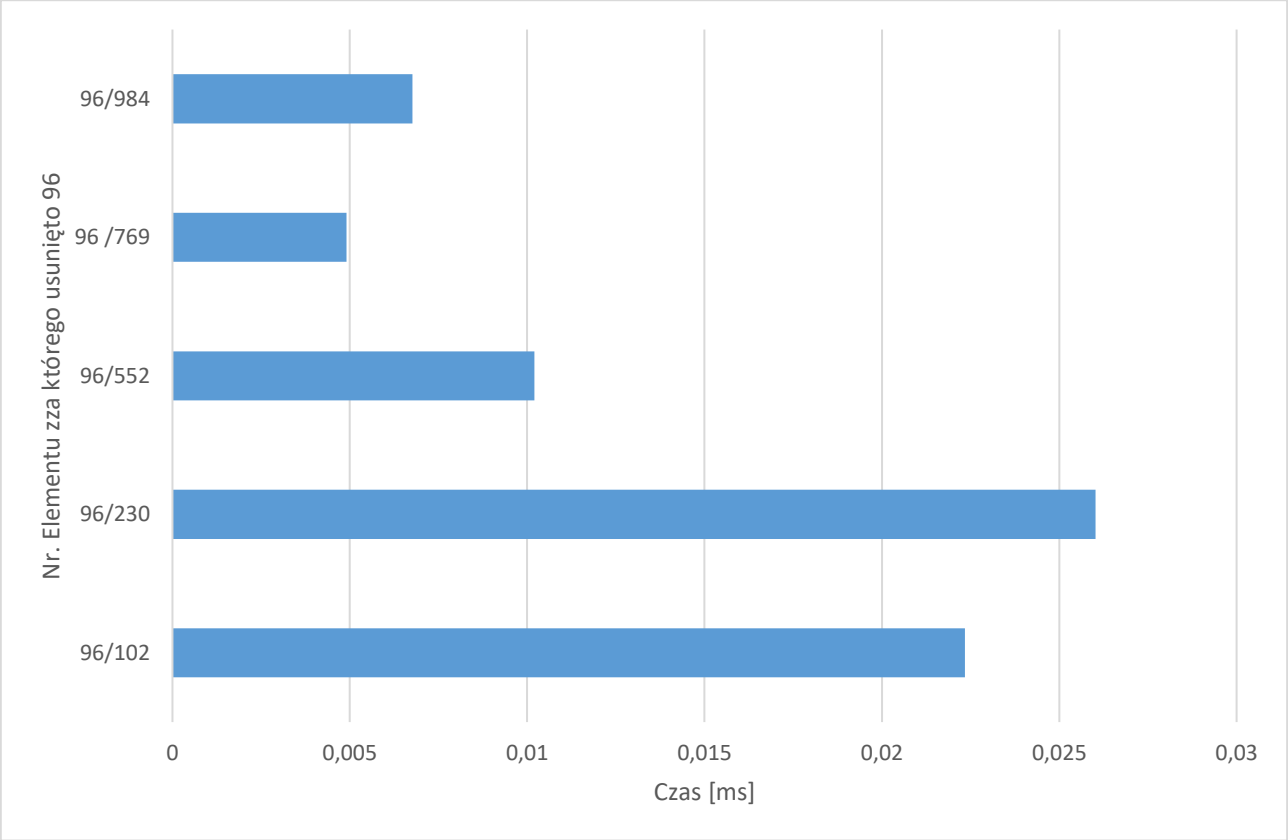


Wykres 12 – Czas zmierzony podczas procedur dodawania nowego elementu po indeksie do listy.

vi. Usuwanie wybranego elementu po wartości

Lp.	Klucz elementu	Średni czas [ms]
1	96/102	0,02234
2	96/230	0,02602
3	96/552	0,010204
4	96 /769	0,00491
5	96/984	0,006764

Tabela 13 – Usuwanie wybranego elementu (Usuwany element został wcześniej dodany za element po „/”)



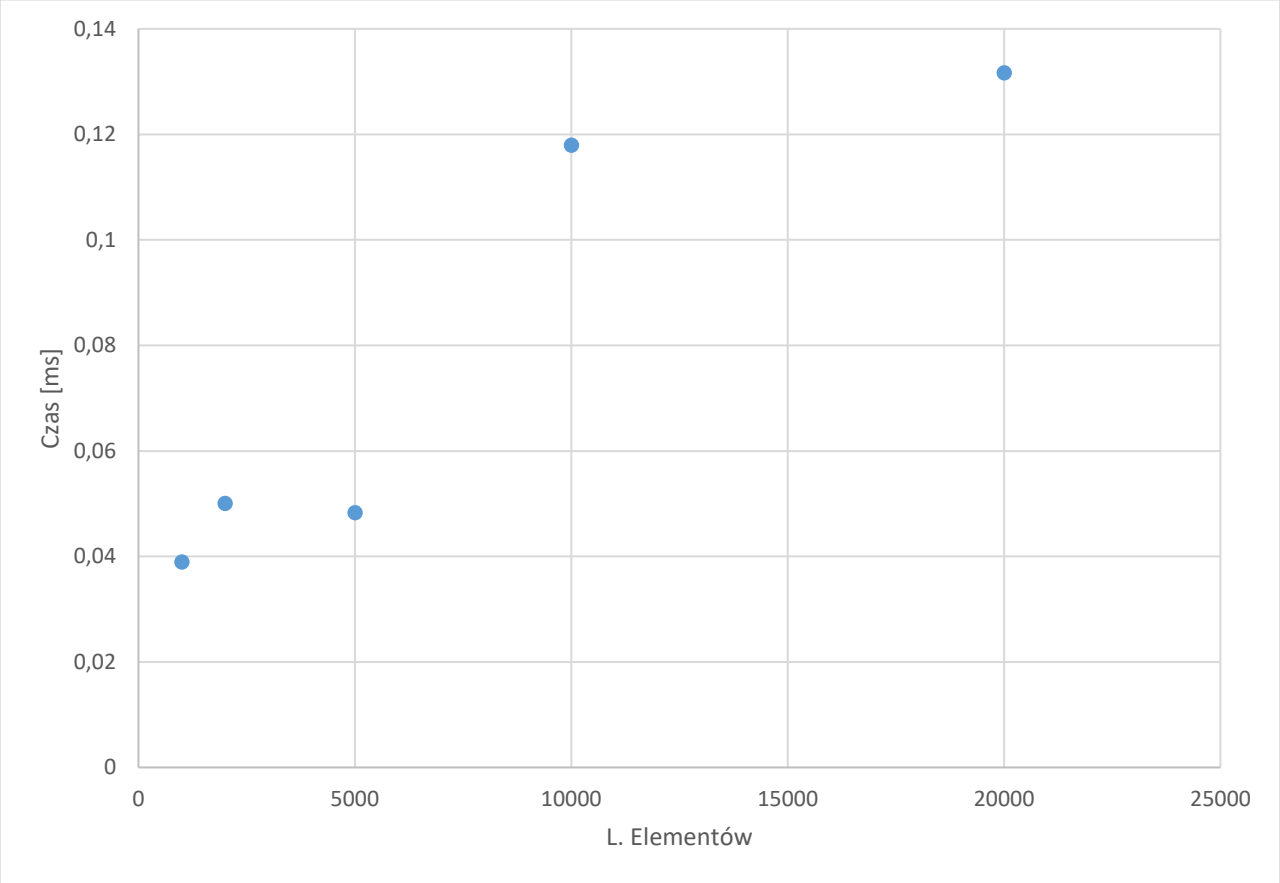
Wykres 13 – Czas zmierzony podczas procedur usuwania konkretnego elementu

W przypadku usuwania elementu z konkretnego miejsca program musi odnaleźć dany element na liście(w przypadku usuwania upewniono się, że dane elementy znajdują się na liście za pomocą funkcji przeszukiwania listy), po czym dodaje go za wskazanym elementem. Jeśli chodzi o dodawanie po indeksie, dodawano element o stałej wartości pod stały indeks, dla wylosowanej za każdym razem innej tablicy naśladując przez to warunki rzeczywiste.

vii. Wyszukiwanie elementu w liście

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,03892
2	2000	0,050036
3	5000	0,04824
4	10000	0,11792
5	20000	0,13166

Tabela 14 – Wyszukiwanie wybranego elementu



Wykres 14 – Zależność czasu potrzebnego na odnalezienie elementu od ilości elementów na liście

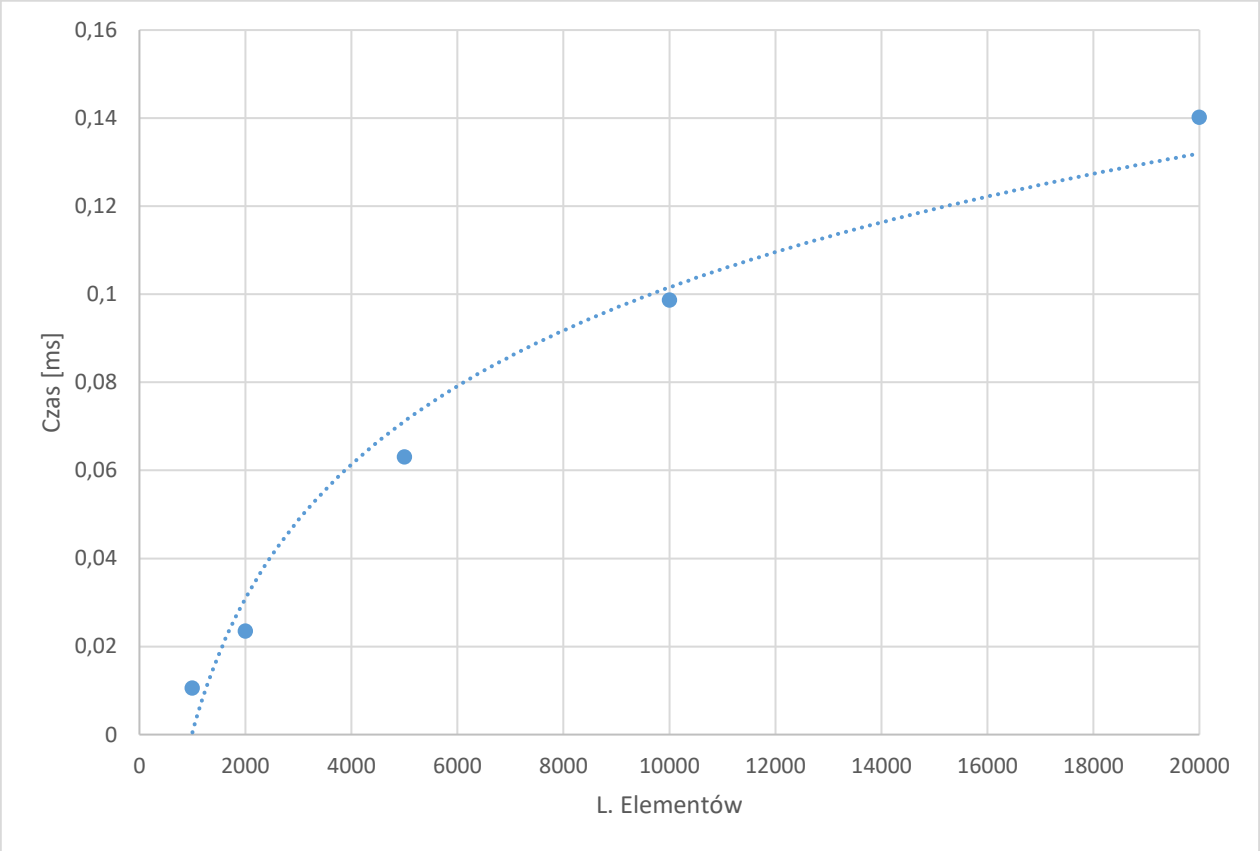
W przypadku przeszukiwania listy czas potrzebny na odnalezienie elementu w dużej mierze zależy od ilości elementów na liście oraz od jego położenia. Im dalej element znajduje się od „głowy" tym przeszukiwanie trwa dłużej.

c) Kopiec

i. Dodawanie elementu

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,010568
2	2000	0,02352
3	5000	0,0631
4	10000	0,0987
5	20000	0,1402

Tabela 15 – Dodawanie elementu do kopca



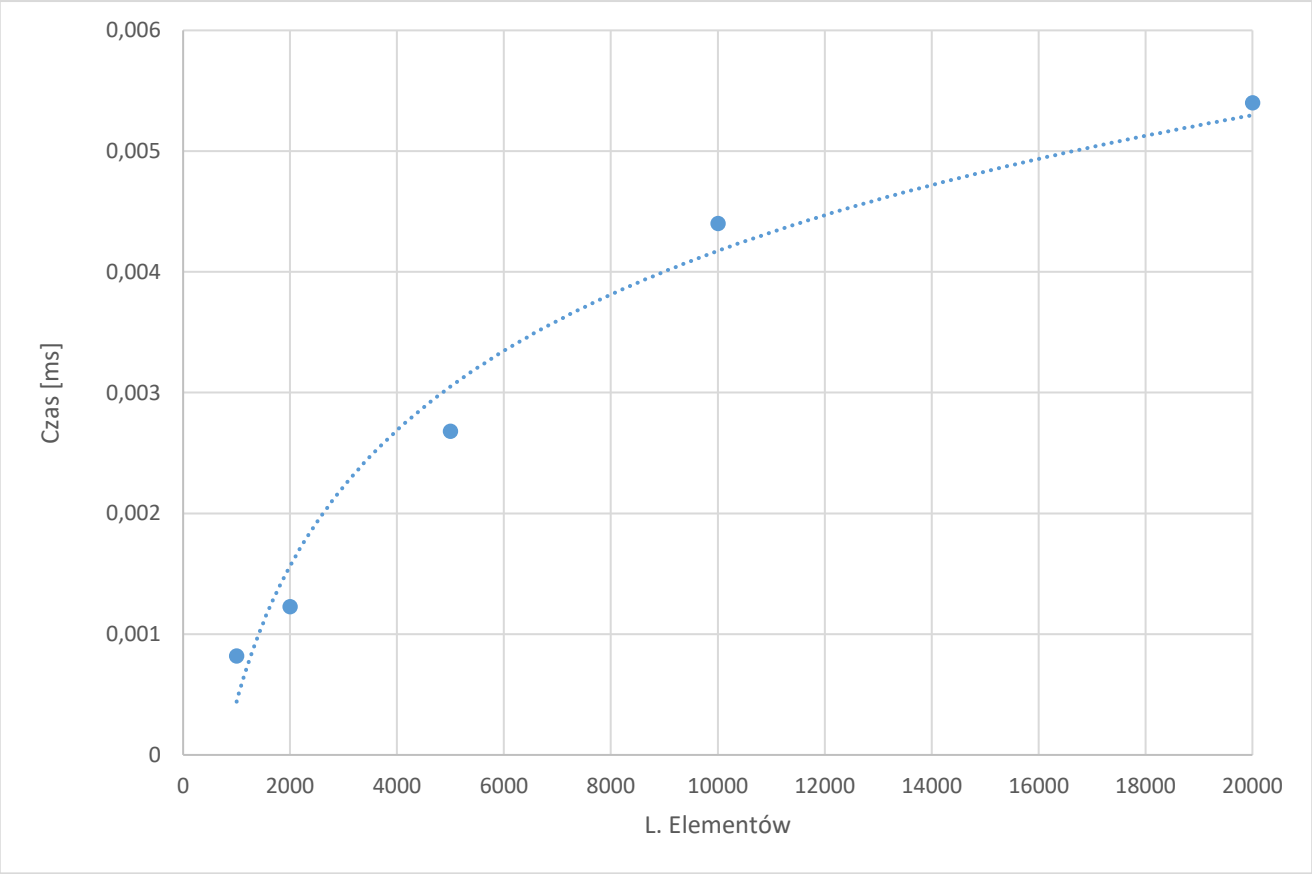
Wykres 15 – Zależność czasu potrzebnego do dodania elementu od ilości elementów w kopcu

Podczas dodawania elementu do kopca czasy zmierzone mogą się różnić. Po dodaniu elementu zostaje uruchomiony algorytm sortowania, który przesuwa nowo dodany element na właściwe miejsce. Zatem czas zależy od tego, o ile poziomów należy przenieść element aby zachować warunek kopca.

ii. Usuwanie elementu

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,0008188
2	2000	0,0012274
3	5000	0,00268
4	10000	0,0044
5	20000	0,0054

Tabela 16 – Usuwanie elementu z kopca

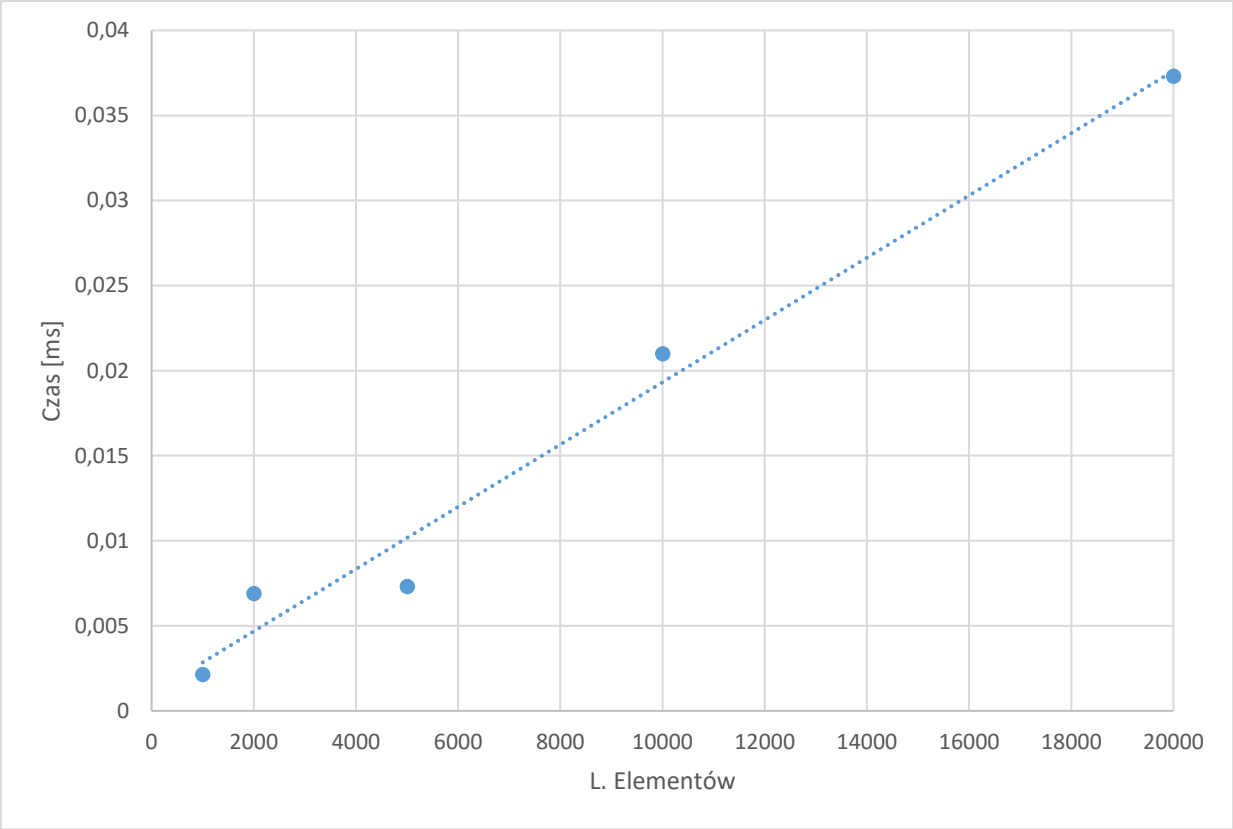


Wykres 16 – Zależność czasu potrzebnego na usunięcie elementu od ilości elementów w kopcu

iii. Wyszukiwanie

Lp.	L. Elementów	Średni czas [ms]
1	1000	0,00213
2	2000	0,00689
3	5000	0,007304
4	10000	0,021
5	20000	0,0373

Tabela 17 – Wyszukiwanie elementu



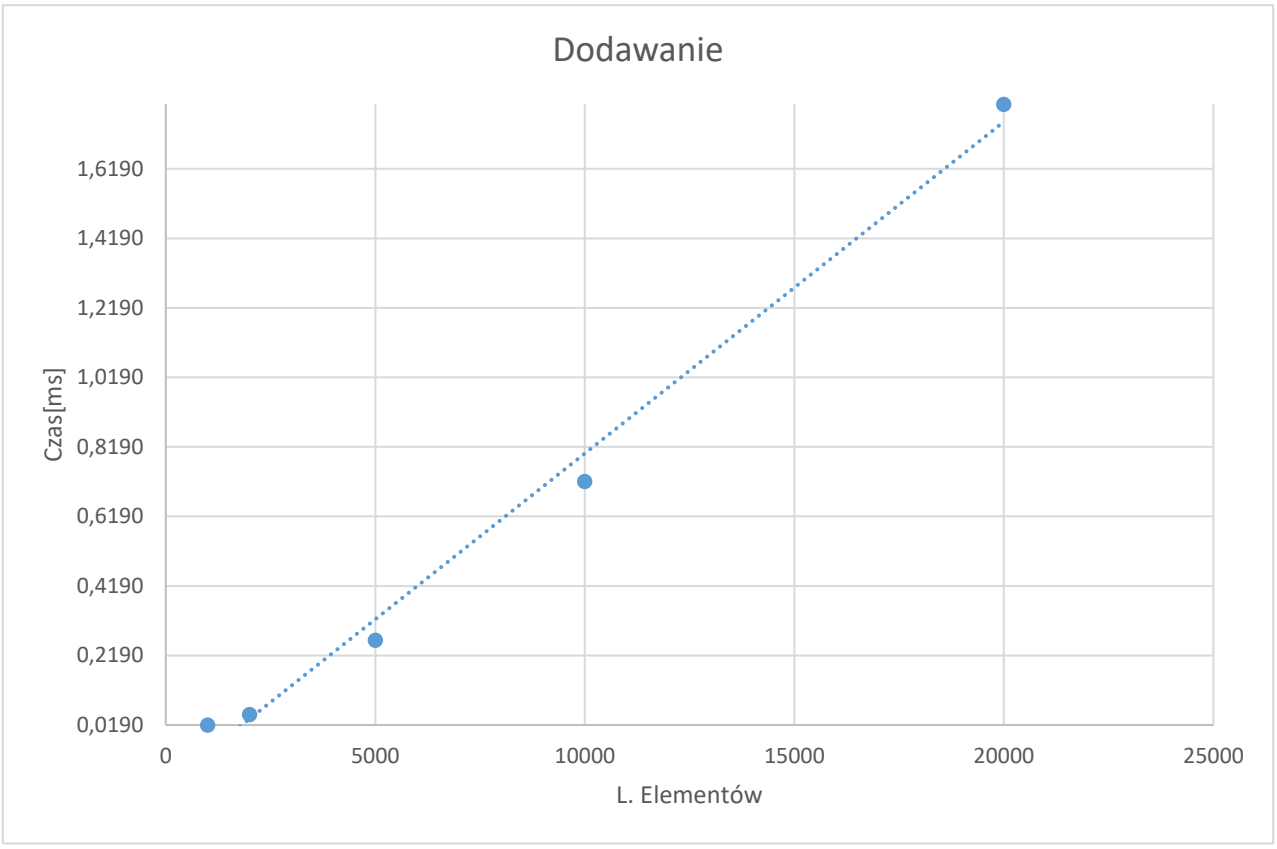
Wykres 17 – Zależność czasowa wyszukiwania elementu od ilości elementów w kopcu

Drzewo BST

iv. Dodawanie Elementu

Lp.	L. Elementów	Czas[ms]
1	1000	0,0195010649
2	2000	0,0494474369
3	5000	0,2626840808
4	10000	0,7201301124
5	20000	1,8041425946

Tabela 18 – Dodawanie elementu do drzewa BST



Wykres 18 – Zależność czasu potrzebnego na dodanie elementu od ilości elementów w drzewie

v. Usuwanie Elementu

Lp.	L. Elementów	Czas[ms]
1	1000	0,0208136365
2	2000	0,0452176007
3	5000	0,1304301485
4	10000	0,3801858917
5	20000	0,9464075999

Tabela 19 – Usuwanie elementu z drzewa

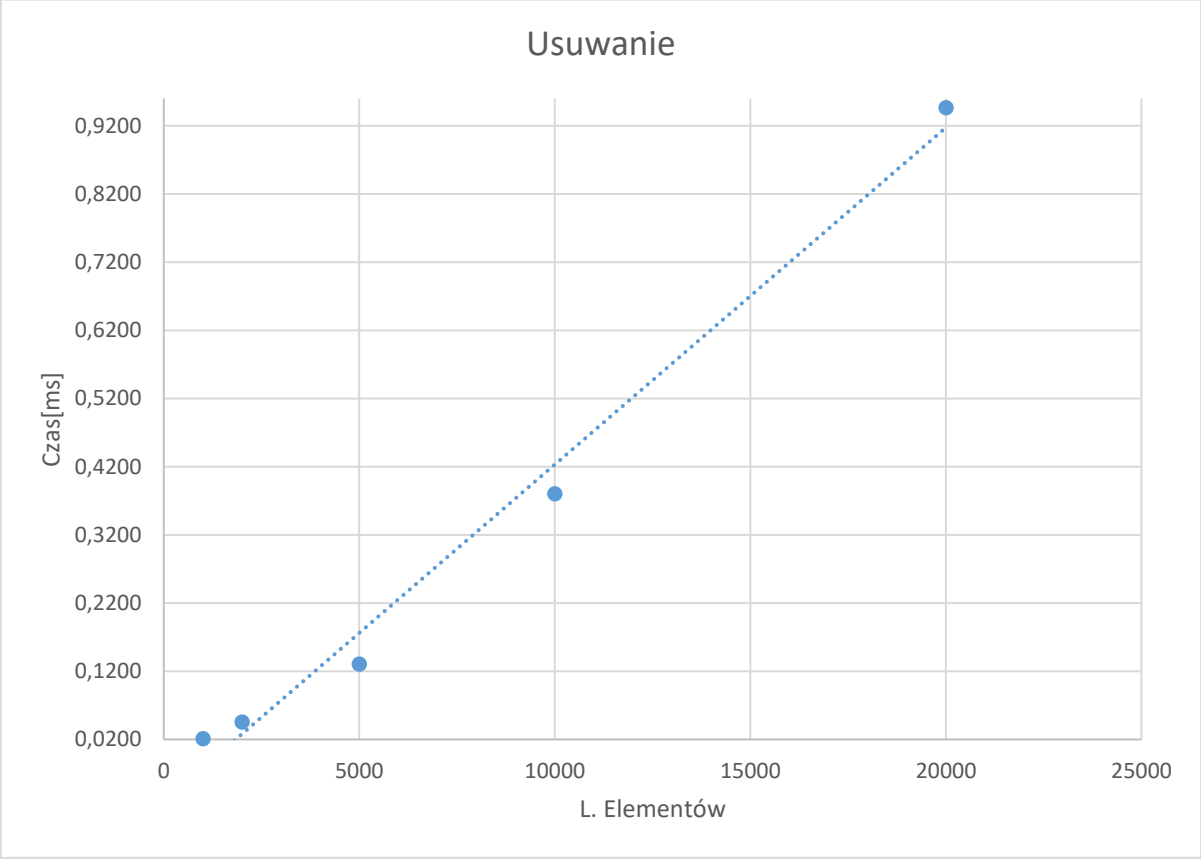
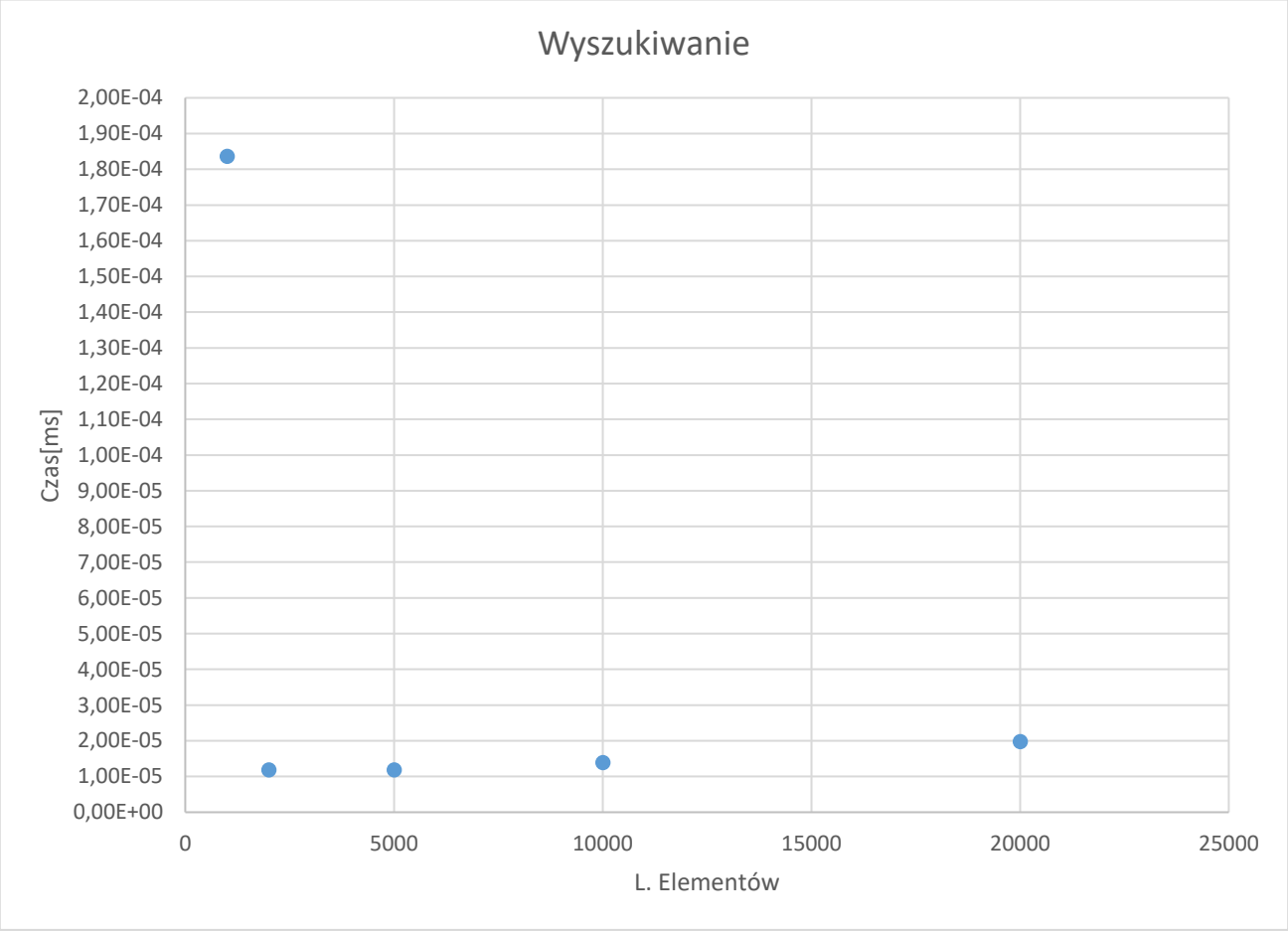


Tabela 19 – Zależność czasu potrzebnego na wyszukanie i usunięcie elementu z drzewa od ilości elementów w drzewie.

vi. Wyszukiwanie

Lp.	L. Elementów	Czas[ms]
1	1000	0,0001835627
2	2000	0,0000118428
3	5000	0,0000118428
4	10000	0,0000138165
5	20000	0,0000197379

Tabela 20 – Wyszukiwanie elementu w drzewie



Wykres 20 – Zależność czasu potrzebnego na wyszukanie elementu w drzewie od ilości elementów

W przypadku wyszukiwania (dla poszczególnych ilości elementów) generowano za każdym razem inne drzewo celem stworzenia warunków zbliżonych do rzeczywistych. Element za każdym razem(jeśli istniał) znajdował się w innym miejscu, stąd duża rozbieżność czasowa.

4. Wnioski

    a) Czasy usunięcia z początku, dodania na początek i wyszukiwania elementu w tablicy rosną wprost proporcjonalnie do liczby elementów, co ilustrują wykresy 1, 2 i 7.
    Operacje usuwania i dodawania na koniec tablicy przeprowadzane są w czasie stałym (wykres 3 i 4). Natomiast czas usunięcia lub wstawienia elementu do wnętrza tablicy jest wprost proporcjonalny do liczby elementów, które leżą na dalszych pozycjach od usuwanego elementu, bądź od miejsca, na które chcemy dodać element (wykresy 5 i 6).
    b) Operacje na elementach krańcowych listy (dodawanie i usuwanie z końca lub początku) mają stałe czasy, co można zaobserwować na wykresach 8, 9,10 i 11. Czas usunięcia lub wstawienia elementu do wnętrza listy uzależniony jest od położenia modyfikowanego elementu na liście.
    c) Logarytmiczną złożoność obliczeniową mają pewne algorytmy używane w kopcu (usuwanie, wstawianie – wykresy 15, 16).

Wartość elementu sama w sobie ma znikomy lub zerowy wpływ na czas jego modyfikacji w strukturze. Natomiast w przypadku struktur takich, jak kopiec czy drzewo BST wartości przechowywanych oraz dodawanych elementów mają znaczenie. W zależności od wartości dodanego elementu, może lub nie musi być konieczne przywrócenie porządku struktury, lub jej zachowanie, dodając lub usuwając z konkretnego miejsca. Podczas testów starano się jednak usuwać i dodawać elementy o bardzo różnych wartościach, aby uśrednione wyniki pokazały faktyczną zależność czasu od liczby elementów.

Dynamiczna alokacja pamięci w przypadku struktur tablicowych jest istotna w prawidłowym działaniu programu. Nie korzystanie z niej skutkuje niestabilnością programu, który po wykonaniu wielu operacji może zakończyć swoje działanie w wyniku błędu pamięci. Jednak korzystanie z relokacji po każdym usunięciu/dodaniu elementu wydłuża czas operacji.

Badane w projekcie struktury mają różne zastosowania w zależności od ich specyficznych cech. Tablice dają się łatwo przetwarzać dzięki bezpośredniemu dostępowi do konkretnego elementu poprzez indeks. Listy służą do reprezentacji zbiorów dynamicznych, takich jak stosy i kolejki, ponieważ nie wymagają ręcznej alokacji pamięci. Kopca używa się przede wszystkim w algorytmie sortowania przez kopcowanie.