Struktury Danych i Złożoność Obliczeniowa

Sprawozdanie	Zadanie projektowe nr 1
Imię i nazwisko, nr indeksu:	Piotr Komarnicki, 264486
Termin zajęć:	Wtorek 15:15 TP
Numer grupy ćwiczeniowej:	K03-37p
Prowadzący kurs:	Mgr inż. Antoni Sterna

1. Wprowadzenie

Celem projektu było własnoręczne zaimplementowanie prostych struktur danych i dostarczenie własnych implementacji operacji pozwalających na dodanie, usuwanie i sprawdzenie czy element znajduje się w strukturze. Struktury, które rozważaliśmy w ramach tego projektu to:

- tablica dynamiczna
- lista dwukierunkowa
- kopiec binarny
- drzewo poszukiwań binarnych (BST)

W drugiej części projektu musieliśmy przeprowadzić analizę czasową stworzonych struktur, i operacji na nich wykonywanych, aby móc porównać je z teoretycznymi wartościami, które prezentują się następująco:

Dla tablicy dynamicznej:

- dodawanie elementu – O(n)
- usuwanie elementu – O(n)
- wyszukanie elementu – O(n)

Dla Listy dwukierunkowej:

- dodawanie elementu na początek lub koniec – O(1)
- usuwanie elementu z początku lub z końca – O(1)
- wyszukanie elementu – O(n)
- dodanie lub usunięcie elementu na podanym indeksie – O(n)

Dla kopca binarnego:

- dodawanie i usuwanie elementów – O(log n)
- wyszukanie elementu – O(n)

Drzewo BST:

- dodawania i usuwanie – O(log n)
- wyszukiwanie – O(log n)

2. Plan eksperymentu

Struktury zostały zrealizowane w ramach jednego programu napisanego w języku c++.

Zakładamy, że struktury przechowują 4 bajtową liczbę ze znakiem (int).

Wszystkie struktury są dynamicznie alokowane i przy każdej zmianie następuje ewentualne zwolnienie zasobów.

Lista dwukierunkowa posiada pole z rozmiarem obecnej listy, które jest aktualizowane przy każdej zmianie.

Kopiec binarny jest zaimplementowany w wersji bazującej na tablicy dynamicznej, która jest jedną ze struktur stworzonych w ramach tego projektu.

Drzewo poszukiwań binarnych nie posiada metody pozwalającej zbalansować drzewo.

3. Rodzaje testów

A. Testy ręczne

Program pozwala na testowanie zaimplementowanych struktur oraz wywoływanie odpowiednich funkcji przypisanych dla każdej struktury.

W tym celu powstał konsolowy interfejs, który po wybraniu struktury inicjalizuje ją i pozwala wykonywać na niej operacje z listy. Po wykonaniu operacji, która zmienia stan struktury aplikacja wypisuje jej aktualny stan na ekranie.

Lista jest wypisywana dwukrotnie od początku i od końca.

Kopiec binarny oraz drzewo poszukiwań binarnych są przedstawiane w taki sposób aby było lepiej widać ich strukturę.

B. Testy czasowe

Jeśli podamy jako argument wywołania odpowiednio: test1, test2, test3 lub test4 to wywołujemy automatyczne testy dla: tablicy dynamicznej, listy dwukierunkowej, kopca binarnego lub drzewa poszukiwań binarnych.

Testy pomogły mi w łatwy sposób zebrać dane dotyczące czasu wykonywania operacji na strukturach danych, a następnie uzyskane czasy były zapisywane w plikach z rozszerzeniem csv. Każdą operację wykonałem dla 5 różnych ilości danych oraz dodatkowo dla tej samej ilości generowałem 10 różnych populacji.

Test automatyczny polega na:

1. Stworzeniu pustej struktury
2. Wypełnieniu wskazaną ilością losowych danych z zakresu od 0 do MAX_INT
3. Zmierzeniu czasu dla każdej operacji na wygenerowanych danych
4. Zapisaniu danych w pliku z rozszerzeniem csv.

Do generowania losowych danych wykorzystałem bibliotekę <random>.

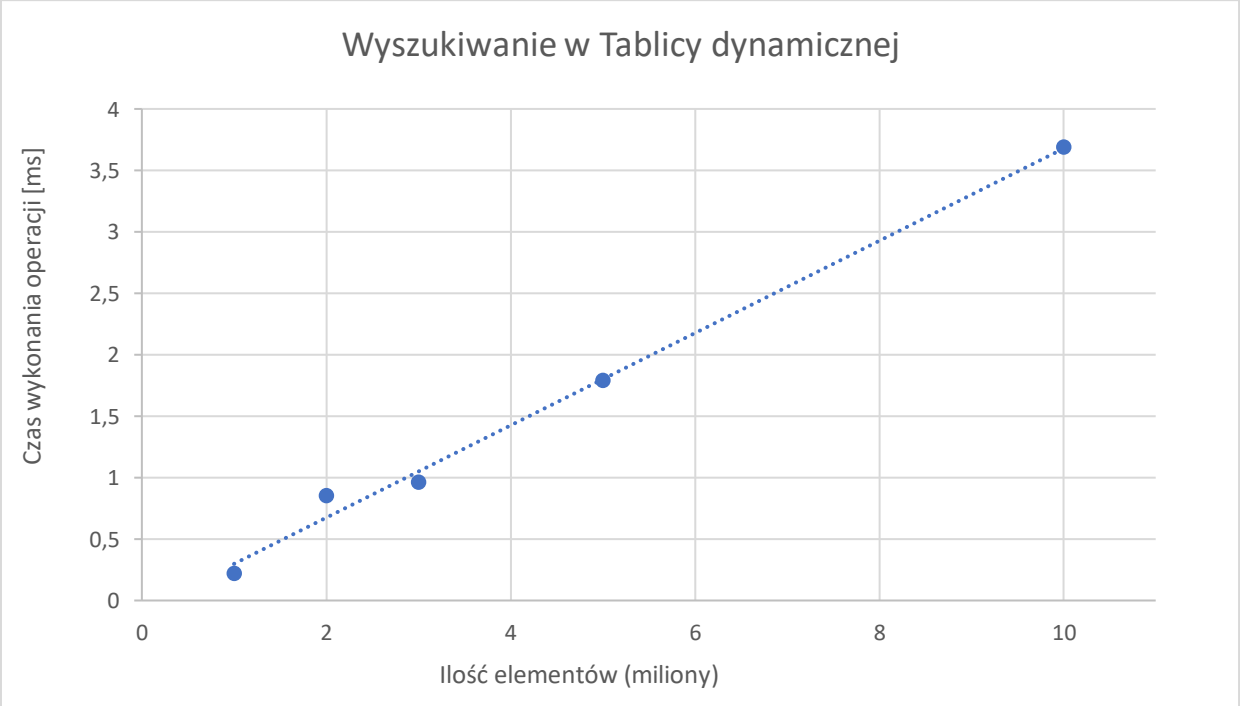
Do pomiarów czasów skorzystałem z std::chrono::high_resolution_clock z biblioteki <chrono>.

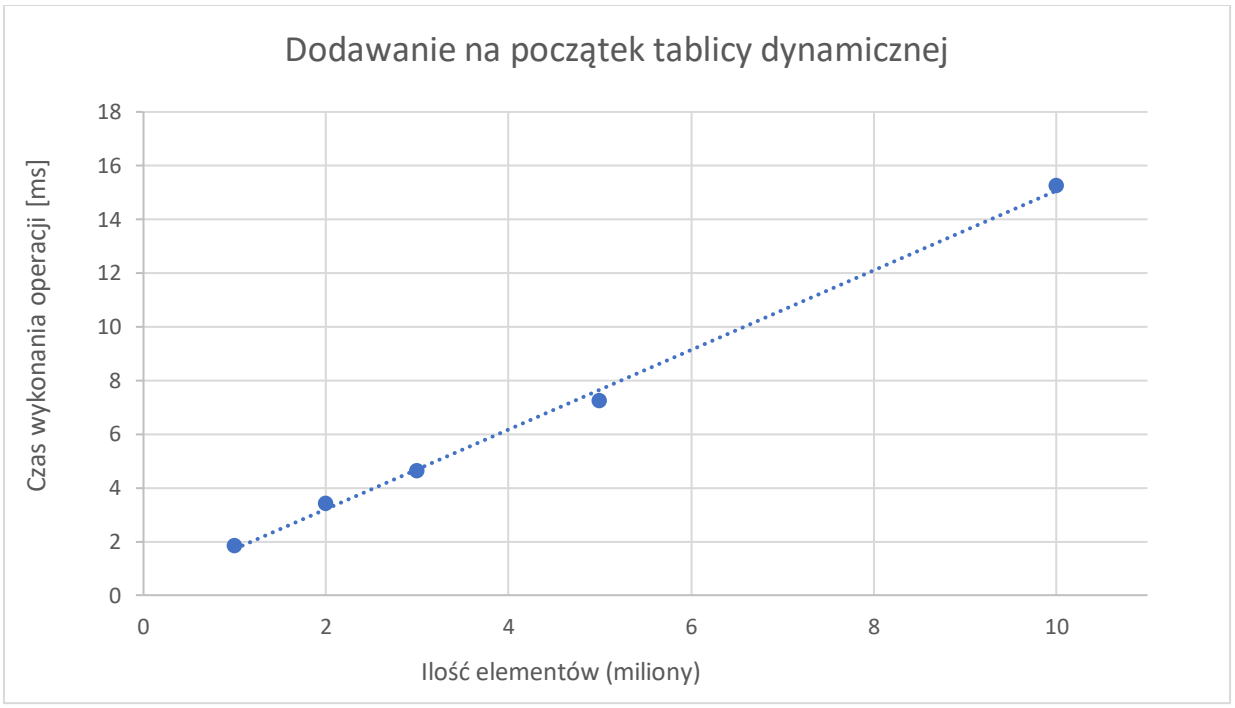
4. Wyniki pomiarów

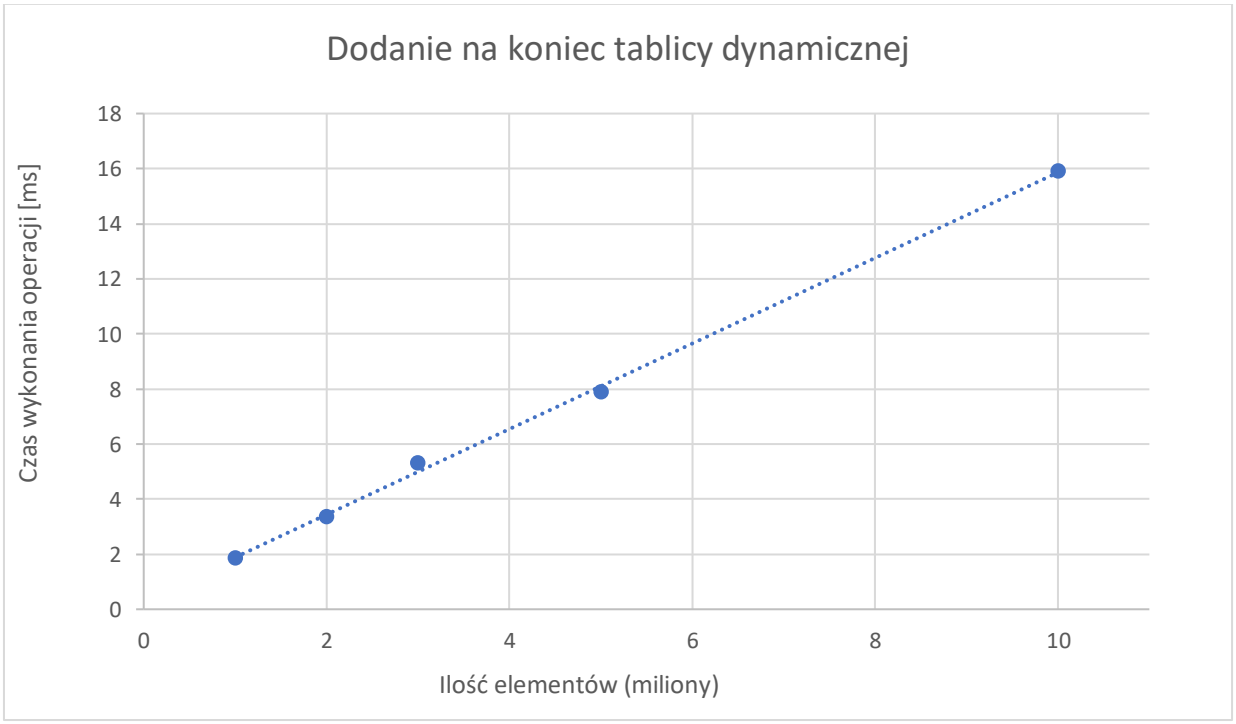
Tablica Dynamiczna:

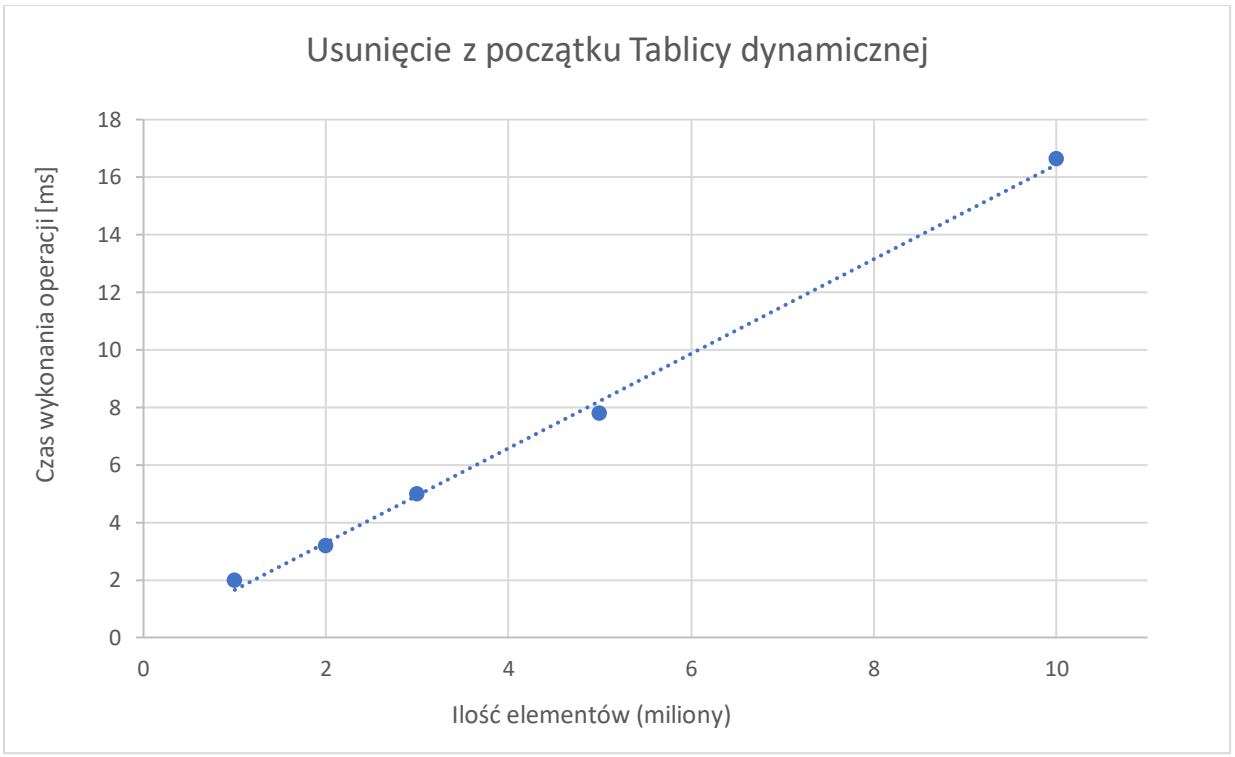
Ilość danych	search	Add_to_beginning	Add_to_back	Add	Delete_from_beginning	Delete_from_back	Delete_element
	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]
1 000 000	0,22	1,84	1,84	1,95	1,97	2,11	1,89
2 000 000	0,85	3,40	3,35	3,43	3,17	3,36	3,23
3 000 000	0,96	4,62	5,30	5,07	4,98	4,91	4,90
5 000 000	1,77	7,22	7,90	8,45	7,78	7,73	8,12
10 000 000	3,68	15,25	15,90	16,89	16,63	17,03	16,26

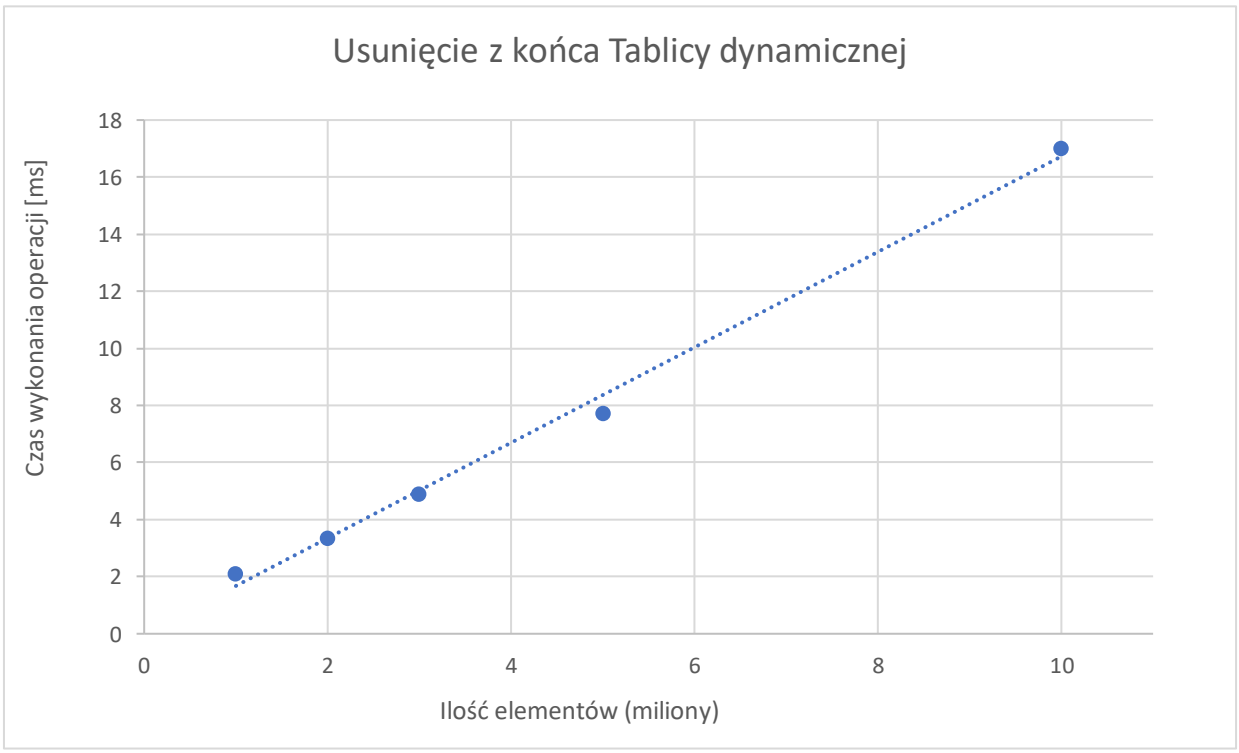
Wykresy:

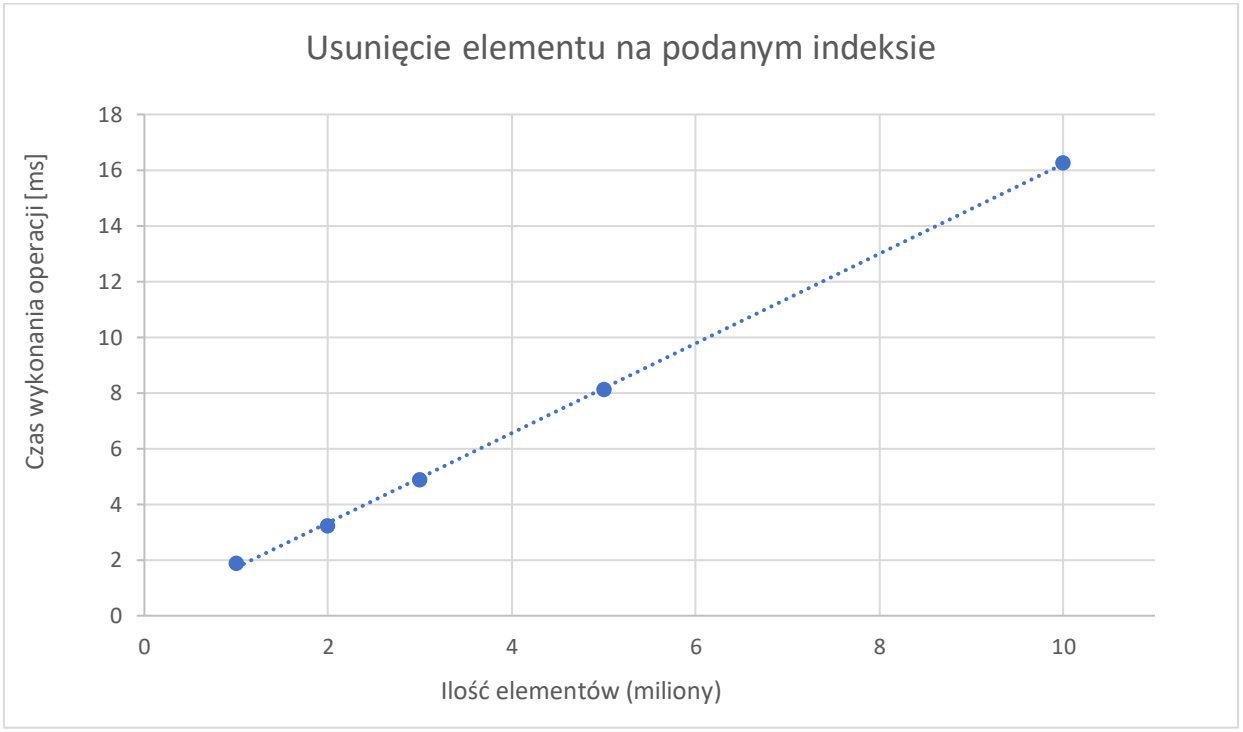



Dodawanie na początek tablicy dynamicznej
Czas wykonania operacji [ms]
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
Ilość elementów (miliony)


Dodanie na koniec tablicy dynamicznej
Czas wykonania operacji [ms]
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
Ilość elementów (miliony)


Usunięcie z początku Tablicy dynamicznej
Czas wykonania operacji [ms]
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
Ilość elementów (miliony)

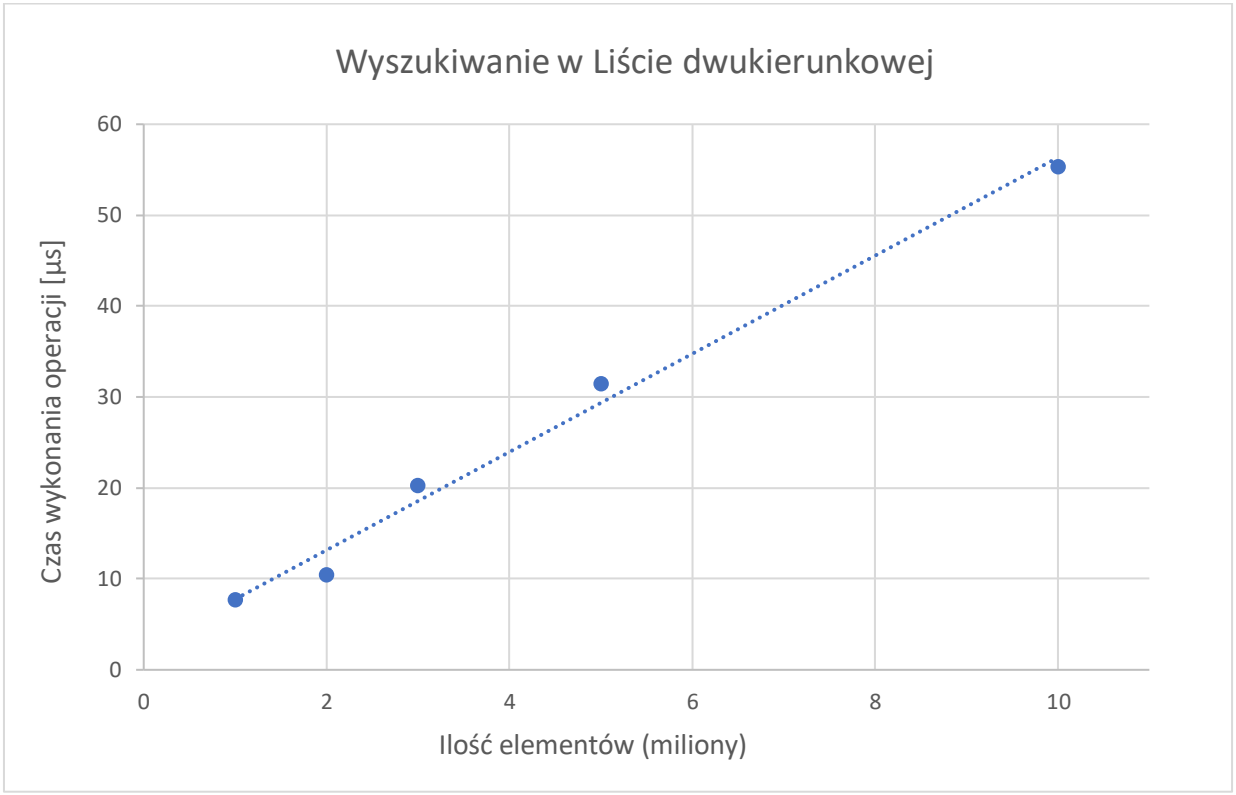

Usunięcie z końca Tablicy dynamicznej
Czas wykonania operacji [ms]
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
Ilość elementów (miliony)

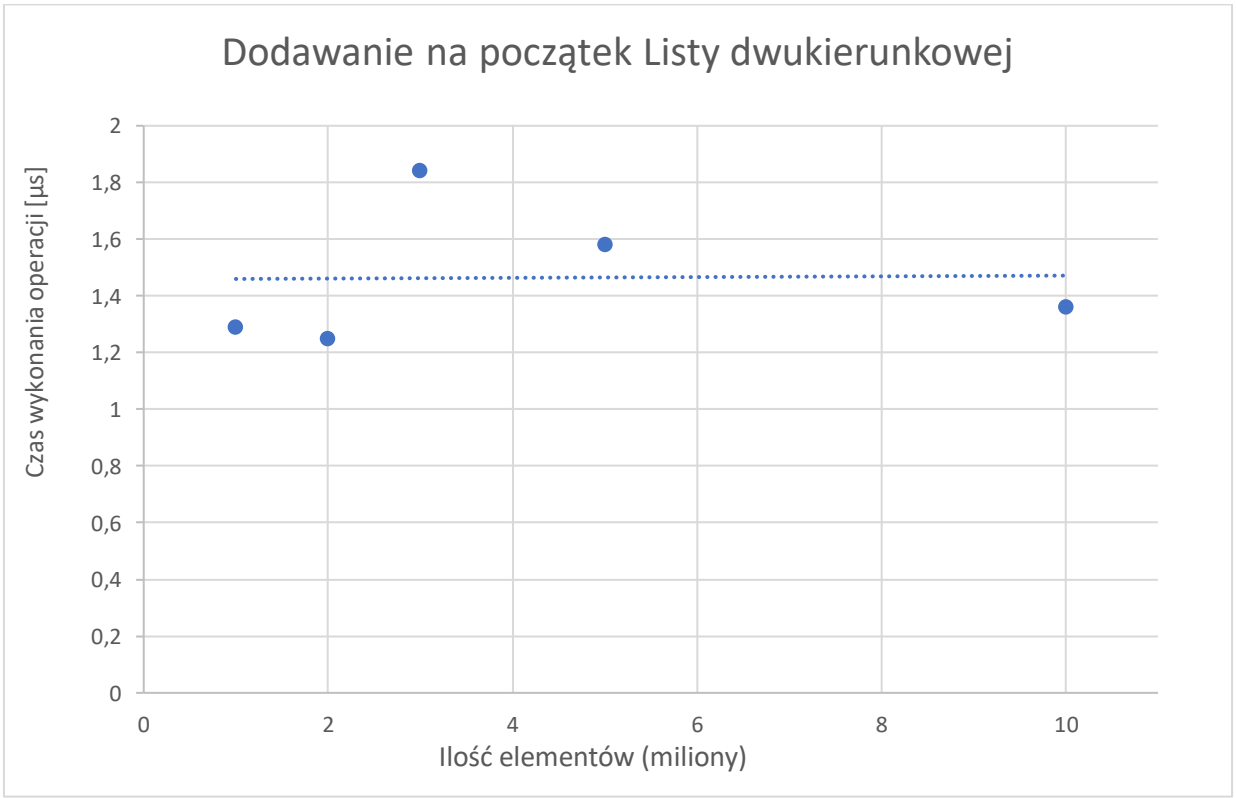


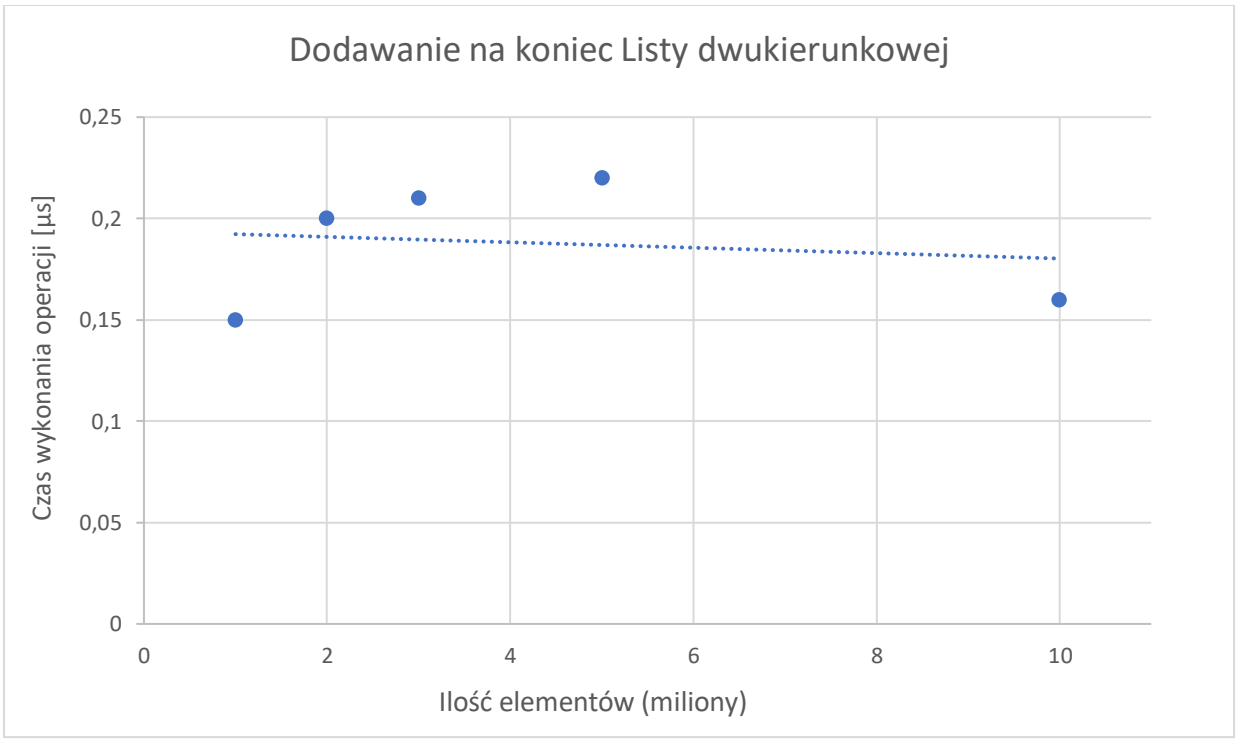
Lista dwukierunkowa:

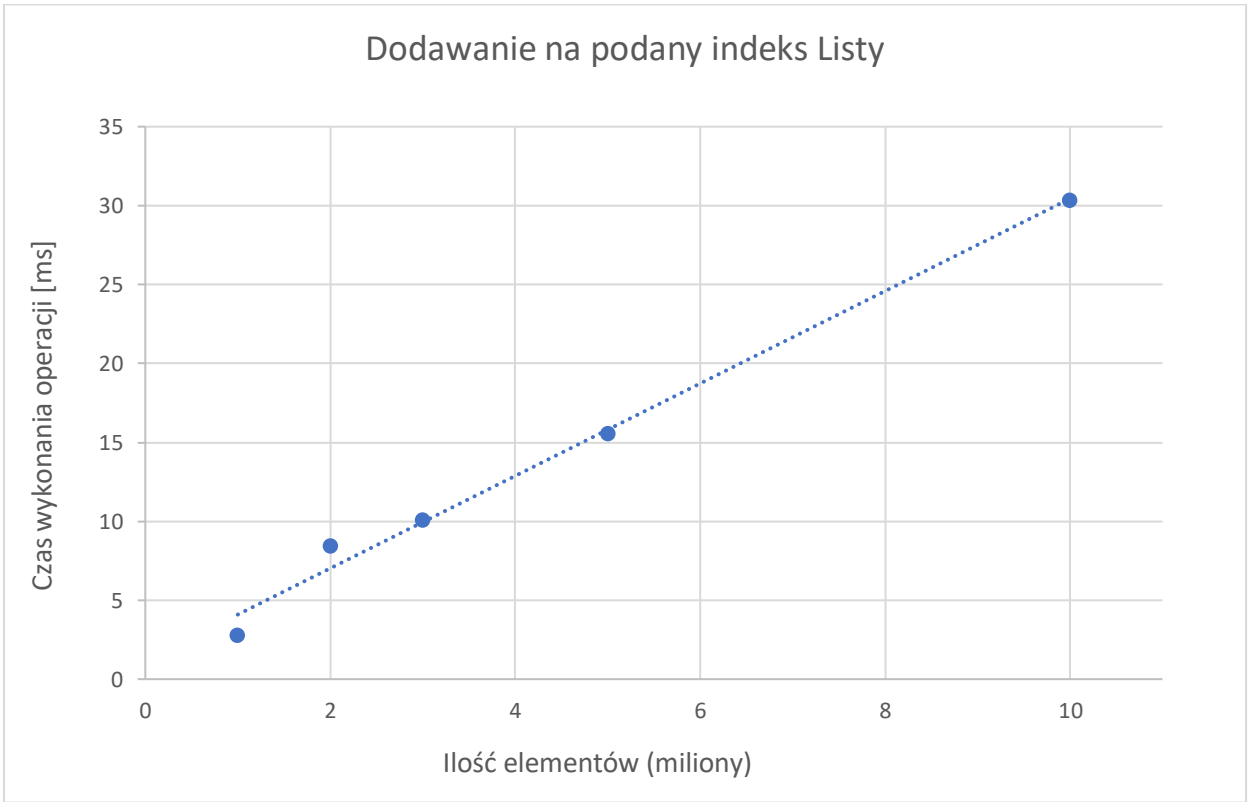
Ilość danych	search	Add_to_beginning	Add_to_end	Add_at_index	Delete_from_beginning	Delete_from_end	Delete_at_index
	[ms]	[µs]	[µs]	[ms]	[µs]	[µs]	[ms]
1 000 000	7,69	1,29	0,15	2,83	0,31	0,72	3,52
2 000 000	10,45	1,25	0,2	8,45	0,30	0,91	6,79
3 000 000	20,26	1,84	0,21	10,11	0,32	1,1	11,27
5 000 000	31,46	1,58	0,22	15,57	0,27	0,95	25,40
10 000 000	55,33	1,36	0,16	30,35	0,27	0,95	35,51

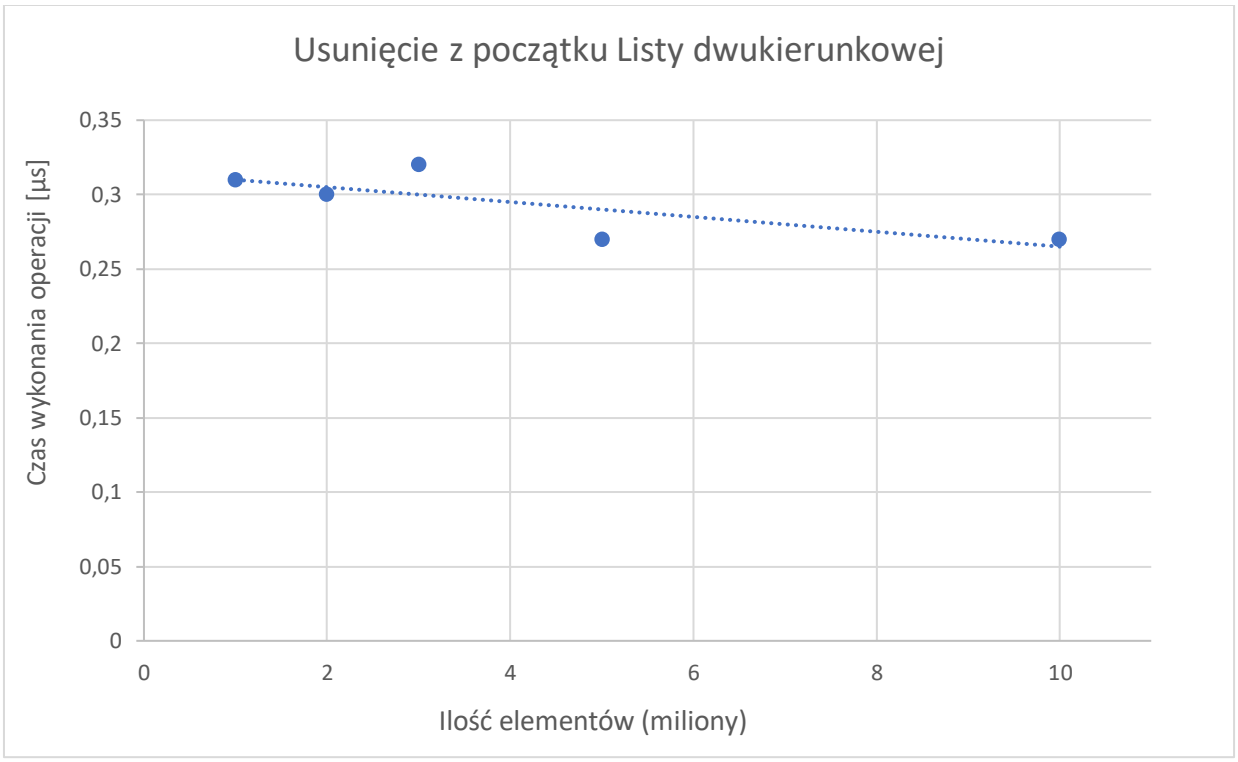
Wykresy:

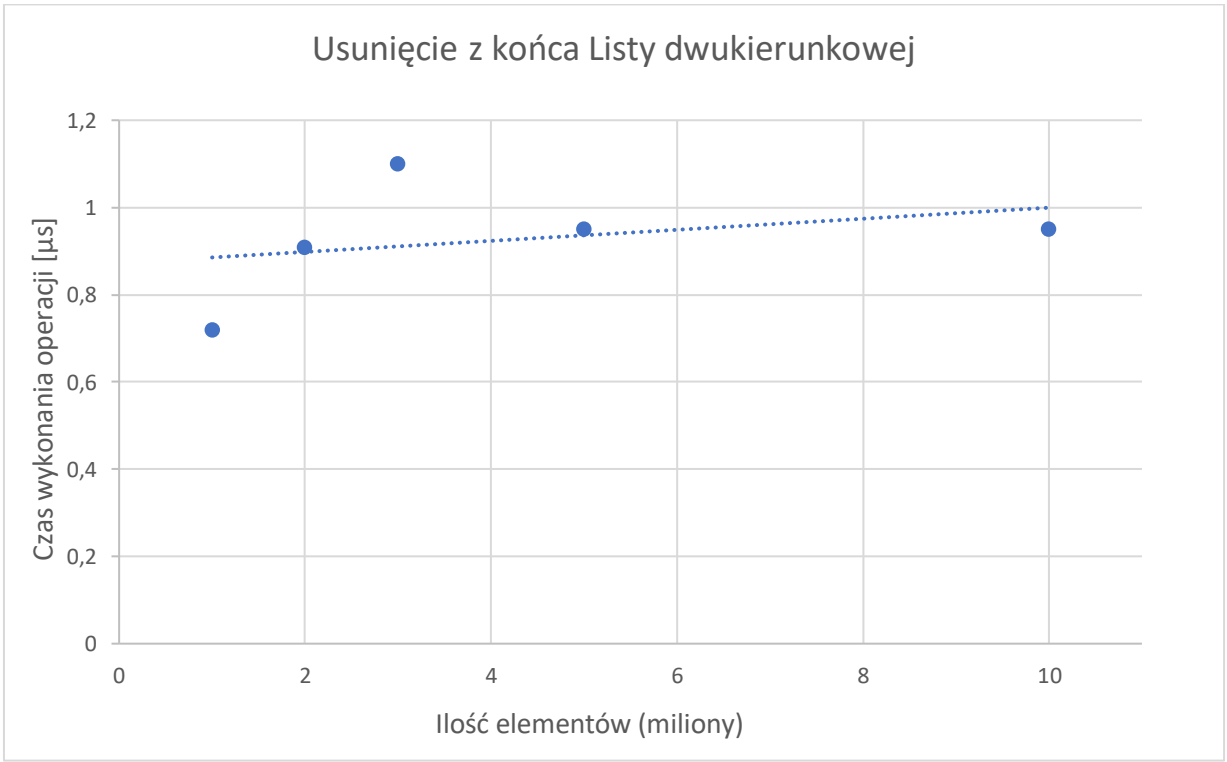

Wyszukiwanie w Liście dwukierunkowej
Czas wykonania operacji [µs]
60
50
40
30
20
10
0
0
2
4
6
8
10
Ilość elementów (miliony)

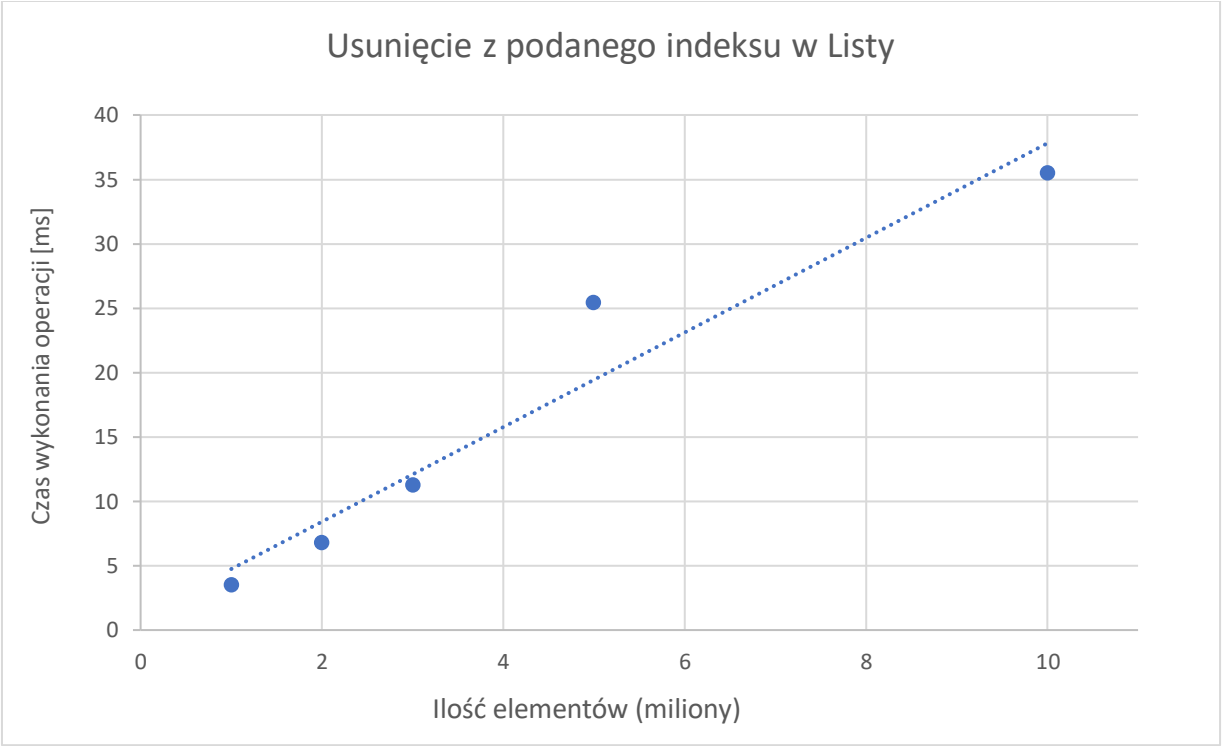

Dodawanie na początek Listy dwukierunkowej
Czas wykonania operacji [µs]
2
1,8
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
0
2
4
6
8
10
Ilość elementów (miliony)


Dodawanie na koniec Listy dwukierunkowej
Czas wykonania operacji [µs]
0,25
0,2
0,15
0,1
0,05
0
0
2
4
6
8
10
Ilość elementów (miliony)


Dodawanie na podany indeks Listy
Czas wykonania operacji [ms]
35
30
25
20
15
10
5
0
0
2
4
6
8
10
Ilość elementów (miliony)


Usunięcie z początku Listy dwukierunkowej
Czas wykonania operacji [µs]
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
2
4
6
8
10
Ilość elementów (miliony)

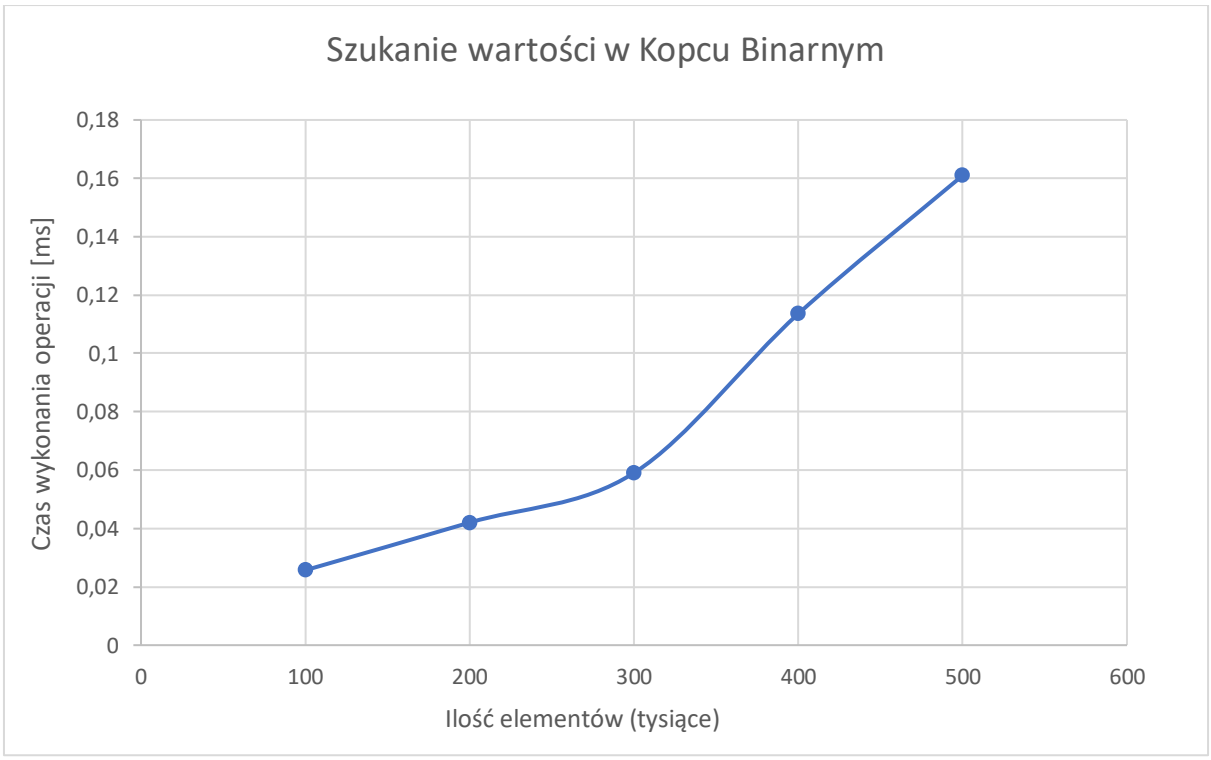

Usunięcie z końca Listy dwukierunkowej
Czas wykonania operacji [µs]
1,2
1
0,8
0,6
0,4
0,2
0
0
2
4
6
8
10
Ilość elementów (miliony)

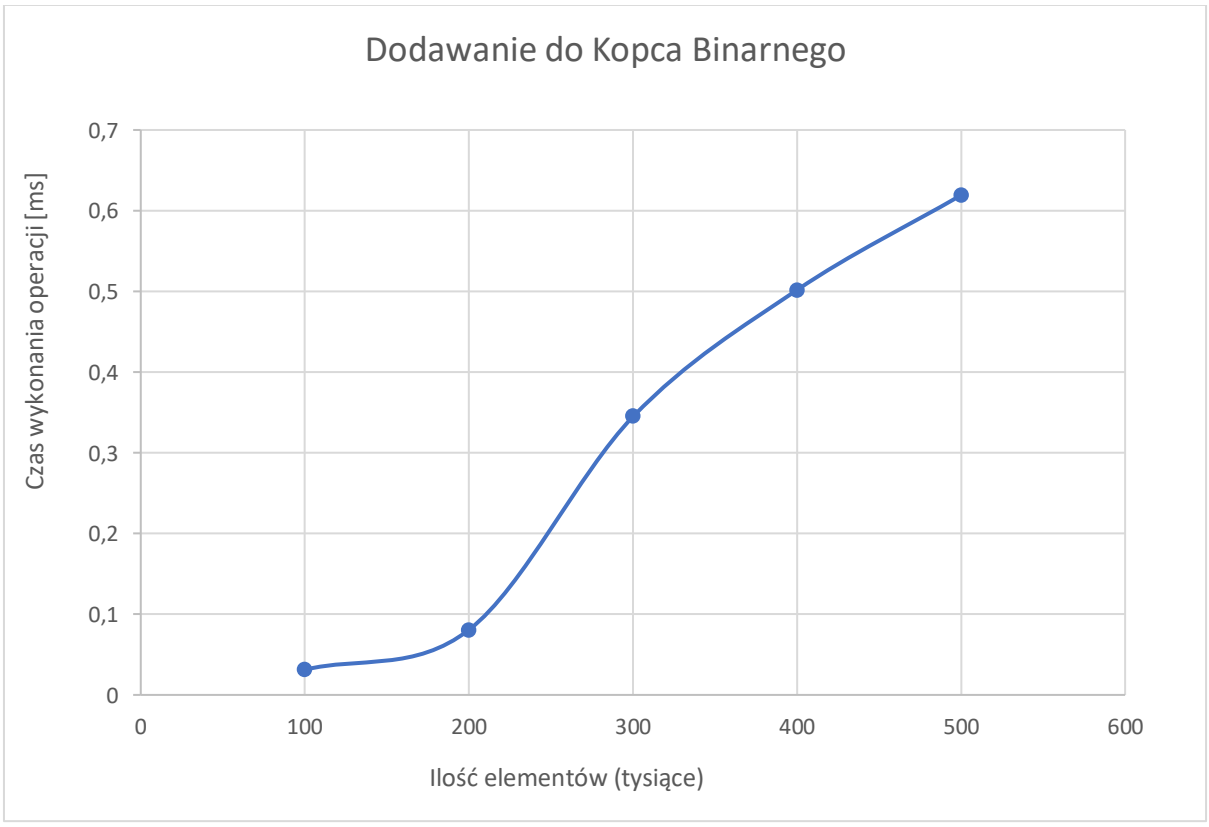


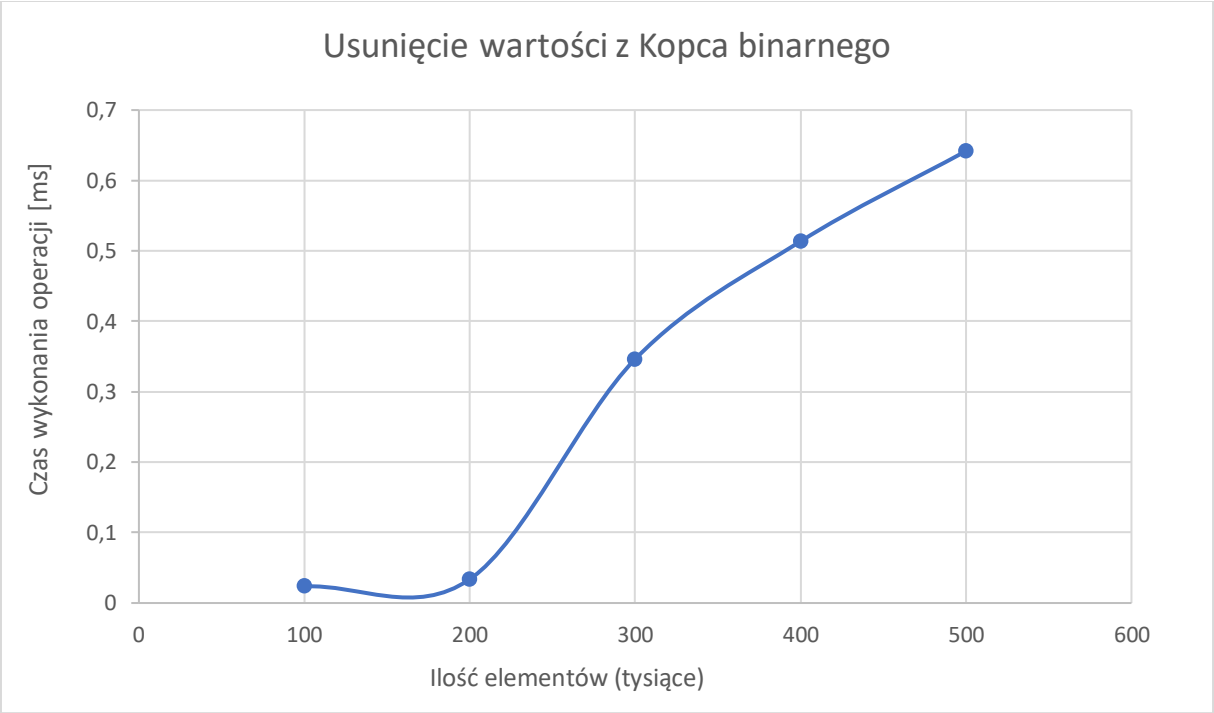
Kopiec Binarny:

Ilość danych	search	add	delete_from_top
	[µs]	[µs]	[µs]
100 000	25,69	31,23	23,64
200 000	42,02	80,57	33,76
300 000	59,03	344,72	346,35
400 000	113,58	501,69	513,63
500 000	160,97	619,17	641,81

Wykresy:


Szukanie wartości w Kopcu Binarnym
Czas wykonania operacji [ms]
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0
100
200
300
400
500
600
Ilość elementów (tysiące)

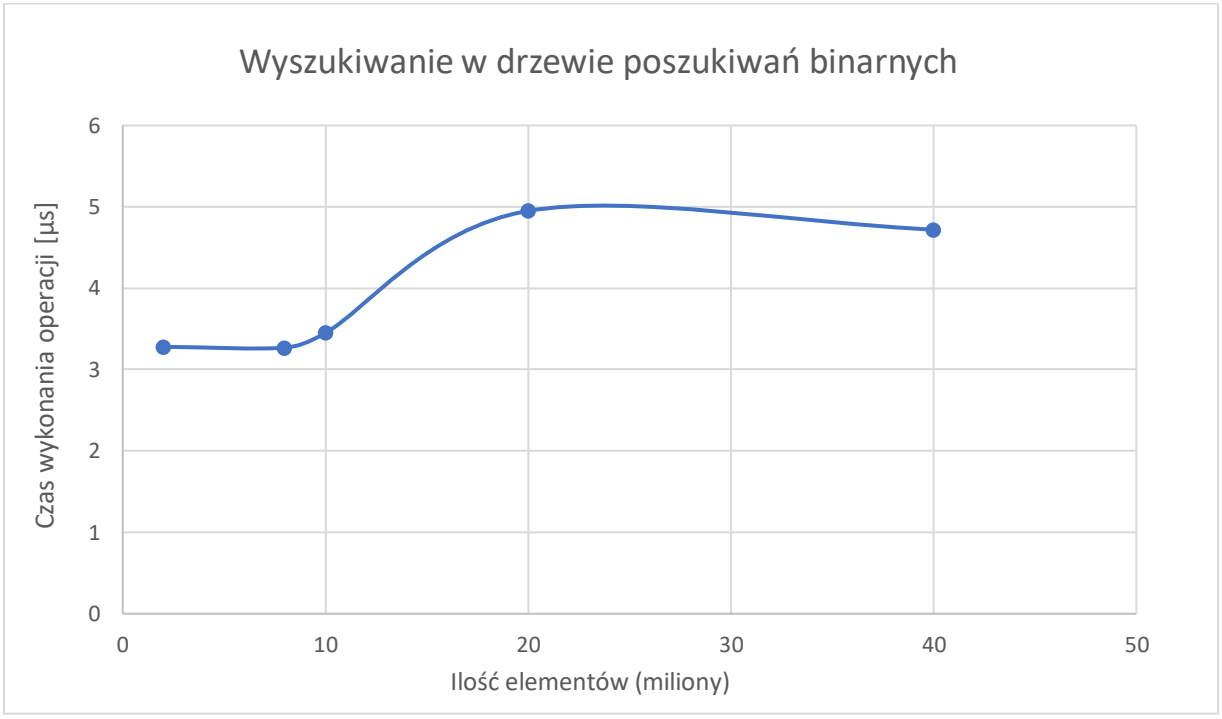

Dodawanie do Kopca Binarnego
Czas wykonania operacji [ms]
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
Ilość elementów (tysiące)

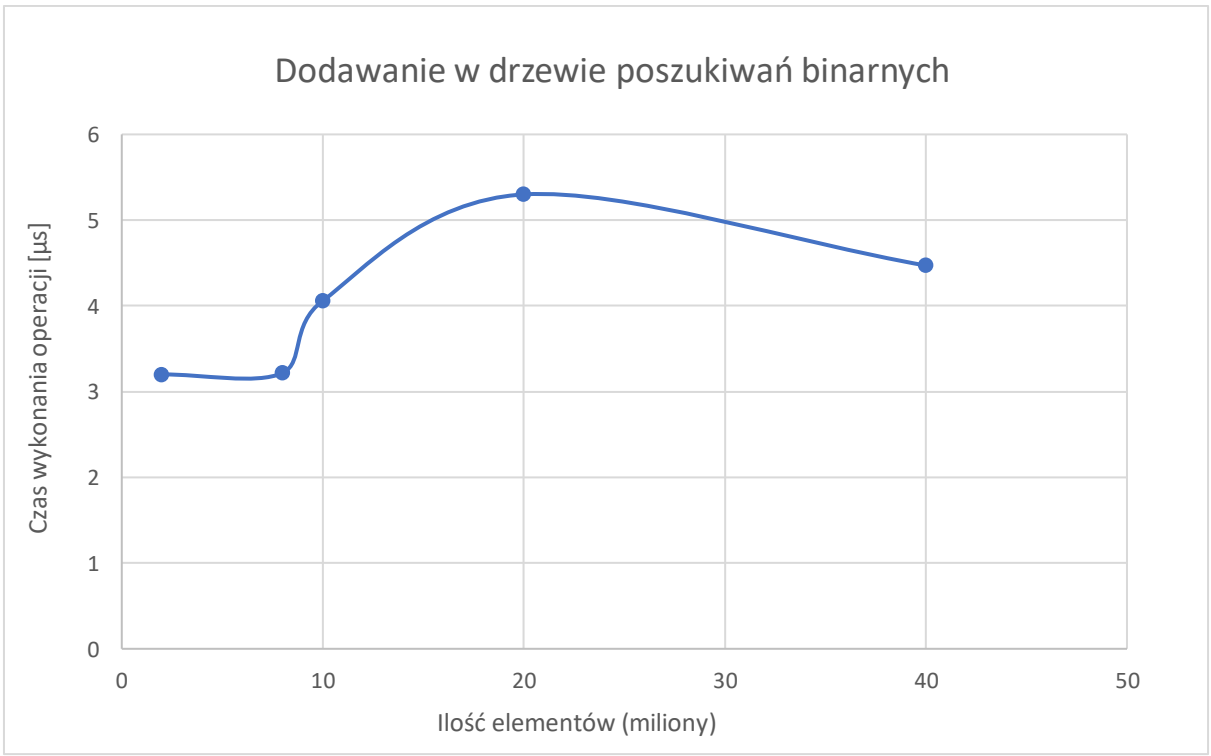


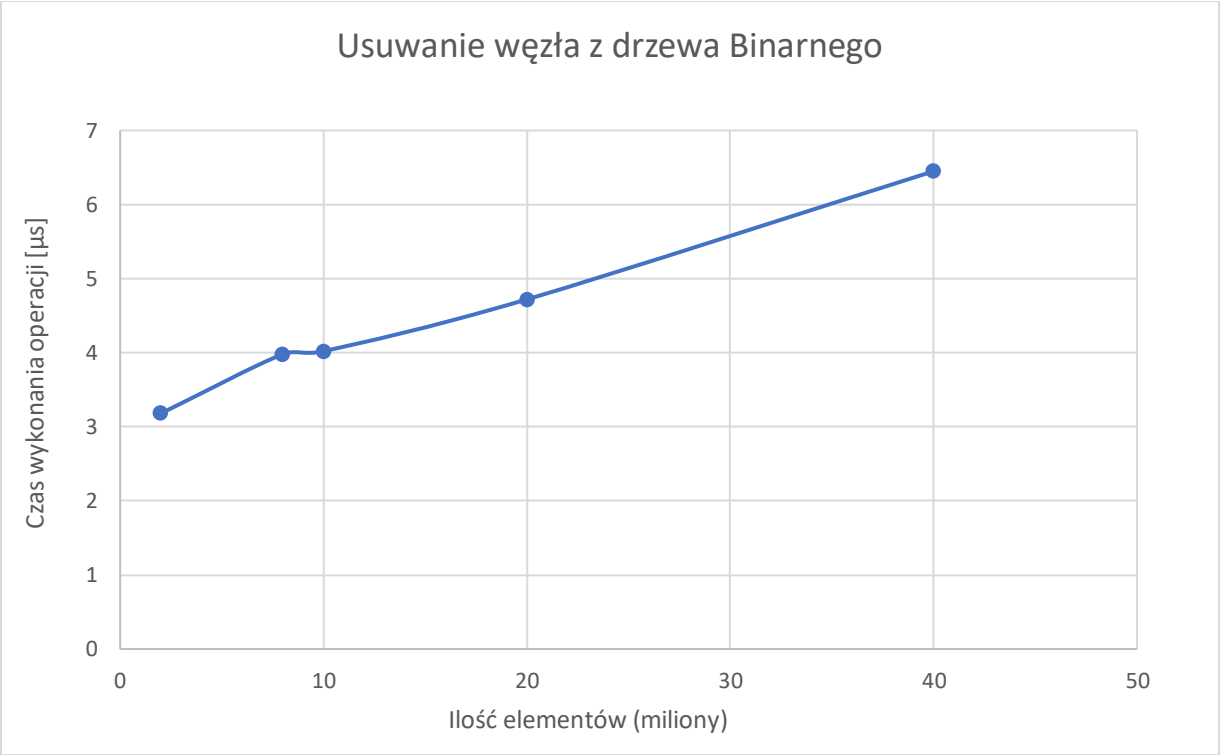
Drzewo poszukiwań binarnych:

Ilość danych	search	Add	Add_to_back
	[µs]	[µs]	[µs]
2 000 000	3,28	3,20	3,18
8 000 000	3,27	3,22	3,98
10 000 000	3,45	4,06	4,02
20 000 000	4,95	5,30	4,72
40 000 000	4,72	4,47	6,45

Wykresy:


Wyszukiwanie w drzewie poszukiwań binarnych
Czas wykonania operacji [µs]
0
1
2
3
4
5
6
0
10
20
30
40
50
Ilość elementów (miliony)


Dodawanie w drzewie poszukiwań binarnych
Czas wykonania operacji [µs]
0
1
2
3
4
5
6
0
10
20
30
40
50
Ilość elementów (miliony)



5. Wnioski

Wyniki otrzymane dla tablicy dynamicznej i listy dwukierunkowej pokrywają się z założeniami teoretycznymi. Lista jest idealna do operacji dodawania i usuwania na krańcach, ale poruszanie się po liście zajmuje dużo czasu w porównaniu z tablicą.

Wyniki otrzymane dla kopca binarnego mogą się trochę różnić od zakładanych, ponieważ korzystamy z tablicy dynamicznej, która tworzy nową tablicę za każdym razem gdy dodajemy lub usuwamy element.

Rozbieżności w uzyskanych wynikach czasowych dla operacji na drzewie poszukiwań binarnych mogą wynikać z braku metody balansującej drzewo co sprawia, że struktura drzewa może się bardzo różnić w zależności od populacji.