

Struktury danych	
<b>Kierunek</b> <i>Informatyczne Systemy Automatyki</i>	<b>Termin</b> <i>środa TP 15<sup>15</sup> – 16<sup>55</sup></i>
<b>Imię, nazwisko, numer albumu</b> <i>Mikołaj Nowak 280082, Nathan Smyrek 280166</i>	<b>Data</b> <i>18.06.2025</i>
<b>Link do projektu</b> <a href="https://github.com/Vexmeritmen/StrukturyDanych3">https://github.com/Vexmeritmen/StrukturyDanych3</a>	



## SPRAWOZDANIE – MINIPROJEKT 3

---

### Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Słownik . . . . .	1
1.2 Tablica mieszająca z adresowaniem otwartym . . . . .	2
1.2.1 Opis struktury . . . . .	2
1.3 Tablica mieszająca z adresowaniem zamkniętym oparta o lista wiązaną .	2
1.3.1 Opis Struktury . . . . .	2
1.4 Tablica mieszająca z adresowaniem zamkniętym oparta o drzewo AVL .	2
1.4.1 Opis Struktury . . . . .	2
1.5 Zastosowania tablic mieszających . . . . .	2
1.6 Teoretyczna złożoność obliczeniowa . . . . .	2
<b>2 Założenia Projektowe</b>	<b>3</b>
2.1 Parametry użytego sprzętu . . . . .	3
2.2 Plan eksperymentu . . . . .	3
<b>3 Badania</b>	<b>3</b>
3.1 Przypadek średni . . . . .	3
3.2 Przypadek pesymistyczny . . . . .	6
<b>4 Wnioski</b>	<b>9</b>
<b>5 Źródła</b>	<b>9</b>

### 1 Wstęp

#### 1.1 Słownik

Słownik jest ADT przechowującym elemtny w parach klucz-wartość.

## 1.2 Tablica mieszająca z adresowaniem otwartym

### 1.2.1 Opis struktury

Zaimplementowana tablica mieszająca z otwartym adresowaniem, jest to struktura, która zwiększa swój rozmiar dwukrotnie za każdym razem kiedy dozwolony poziom współczynnika zajętości zostanie przekroczony.

$$\alpha = 0.6 \geq \frac{n}{m}$$

Gdzie n to ilość elementów, a m obecny rozmiar tablicy.

Tablica wypełniona jest pustymi kubelkami, które zapełniamy poprzez użycie funkcji mieszającej na kluczu pary, która chcemy dodać, zwrócony wynik jest naszym indexem. Jeśli otrzymany index w tablicy jest zajęty przechodzimy do kolejnego, do momentu aż nie znajdziemy wolnego kubelka. Przy zwiększaniu rozmiaru tablicy następuje jej przemieszanie.

Funkcja haszująca:

$$h(x) = x \bmod m$$

Gdzie m to obecny rozmiar tablicy.

## 1.3 Tablica mieszająca z adresowaniem zamkniętym oparta o listą wiązaną

### 1.3.1 Opis Struktury

W każdym kubelku, czyli pod indeksem tablicy mieszającej, znajduje się lista jednokierunkowa przechowująca elementy o tym samym indeksie haszującym. Prostsza implementacja, lepsza od tablicy mieszającej z drzewem AVL w dodawaniu elementów na sam koniec oraz gdy liczba elementów jest niewielka i nie potrzeba szybkiego wyszukiwania.

Funkcja haszująca:

$$h(x) = x \bmod m$$

Gdzie m to rozmiar tablicy.

## 1.4 Tablica mieszająca z adresowaniem zamkniętym oparta o drzewo AVL

### 1.4.1 Opis Struktury

W każdym kubelku znajduje się zrównoważone drzewo AVL, które umożliwia szybsze wyszukiwanie i usuwanie w kubelku niż lista dla dużych ilości danych. Jest to lepsza implementacja, gdy: potrzebujemy szybkiego wyszukiwania w dużym zbiorze danych, dane muszą być przechowywane w kolejności według klucza oraz gdy istnieje ryzyko wielu kolizji. AVL gwarantuje, że wysokość drzewa w kubelku będzie logarytmiczna względem liczby elementów w tym kubelku.

Funkcja haszująca:

$$h(x) = x \bmod m$$

Gdzie m to rozmiar tablicy.

## 1.5 Zastosowania tablic mieszających

- Słowniki
- Zbiory
- Pamięci podręczne
- Indeksy baz danych
- Tablice transpozycji gier

## 1.6 Teoretyczna złożoność obliczeniowa

Tabela 1: Złożoność średnia

Operacja	Otwarte	Lista wiązana	AVL
insert()	$O(1)$	$O(1)$	$O(\log n)$
remove	$O(1)$	$O(1)$	$O(\log n)$

Tabela 2: Złożoność Pesymistyczna

Operacja	Otwarte	Lista wiązana	AVL
insert()	$O(n)$	$O(n)$	$O(\log n)$
remove()	$O(n)$	$O(n)$	$O(\log n)$

## 2 Założenia Projektowe

### 2.1 Parametry użytego sprzętu

- System operacyjny : Linux Mint 21.1 Cinnamon
- Processor : Intel i7-1260P 12-generacji (12 rdzeni, 16 wątków, 3.40-4.70 GHz, 18MB cache)
- Pamięć RAM : 32GB
- Środowisko : Clion 2024.1.3
- Kompilator : g++ 11.4.0

### 2.2 Plan eksperymentu

Przeprowadzona badania dla 11 różnych rozmiar struktur : 5000, 8000, 10000, 20000, 30000, 35000, 42500, 50000, 65000, 80000, 100000. Zostały one wypełnione losowymi liczbami z zakresu -2 147 483 647 do 2 147 483 647. Dla operacji 5000, 10000, 20000, 30000, 35000 zostało wykonane 50 prób, natomiast dla operacji 42500, 50000, 65000, 80000, 100000 mniejszą ilość prób w liczbie 10, po czym wyniki zostały uśrednione.

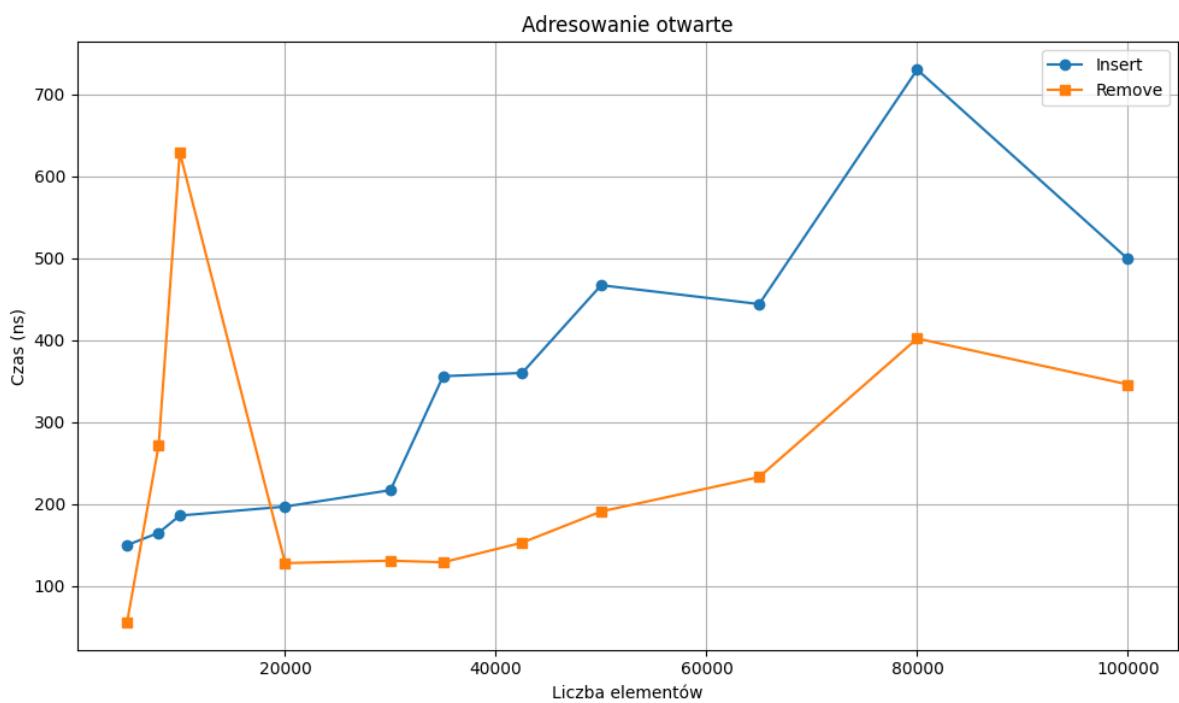
## 3 Badania

### 3.1 Przypadek średni

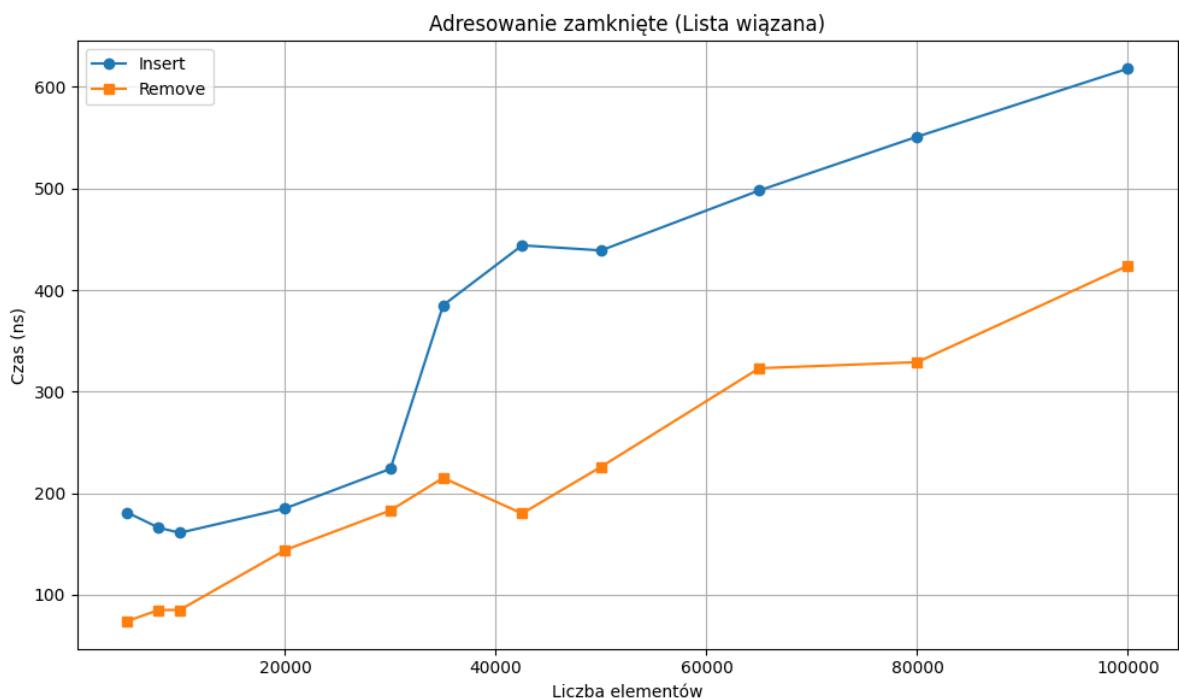
Przypadek średni, losujemy liczby z rozkładu normalnego, niewielka szansa na powtarzające się klucze.

Tabela 3: Przypadek średni dla wszystkich struktur

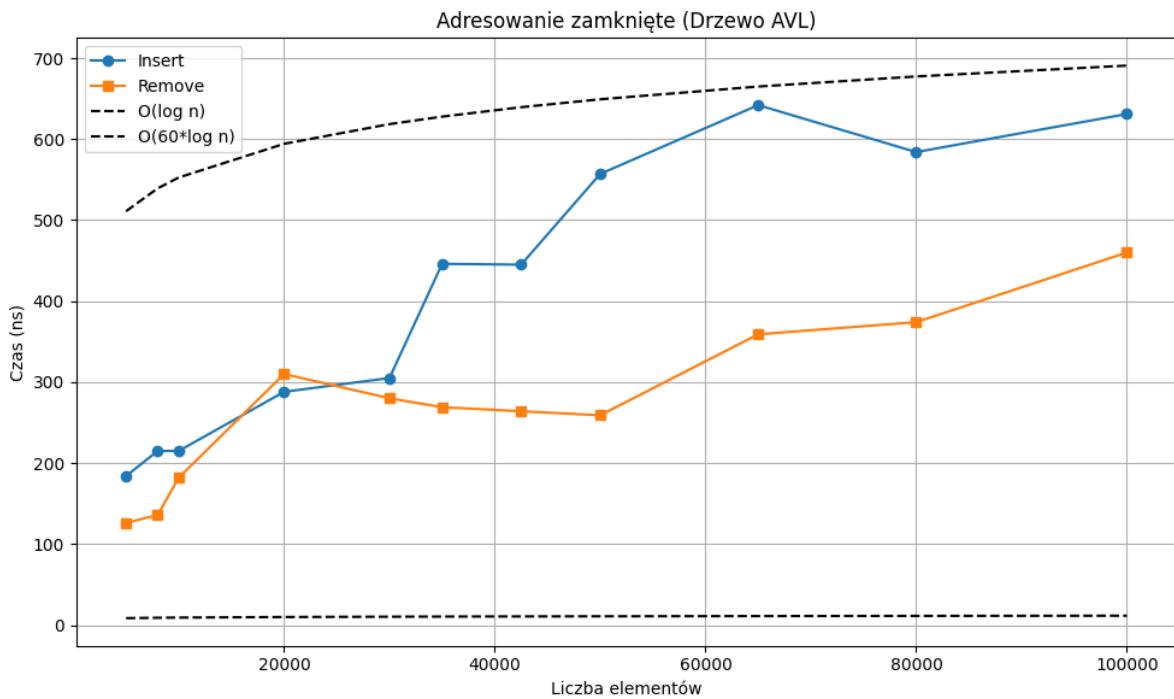
Liczba elementów	Adresowanie zamknięte (drzewo AVL)[ns]		Adresowanie zamknięte (lista wiązana)[ns]		Adresowanie otwarte [ns]	
	Wstawianie	Usuwanie	Wstawianie	Usuwanie	Wstawianie	Usuwanie
5 000	184	126	181	74	150	56
8 000	215	136	166	85	165	272
10 000	215	182	161	85	186	629
20 000	288	310	185	144	197	128
30 000	305	280	224	183	217	131
35 000	446	269	385	215	356	129
42 500	445	264	444	180	360	153
50 000	557	259	439	226	467	191
65 000	642	359	498	323	444	233
80 000	584	374	551	329	730	402
100 000	631	460	618	424	499	346



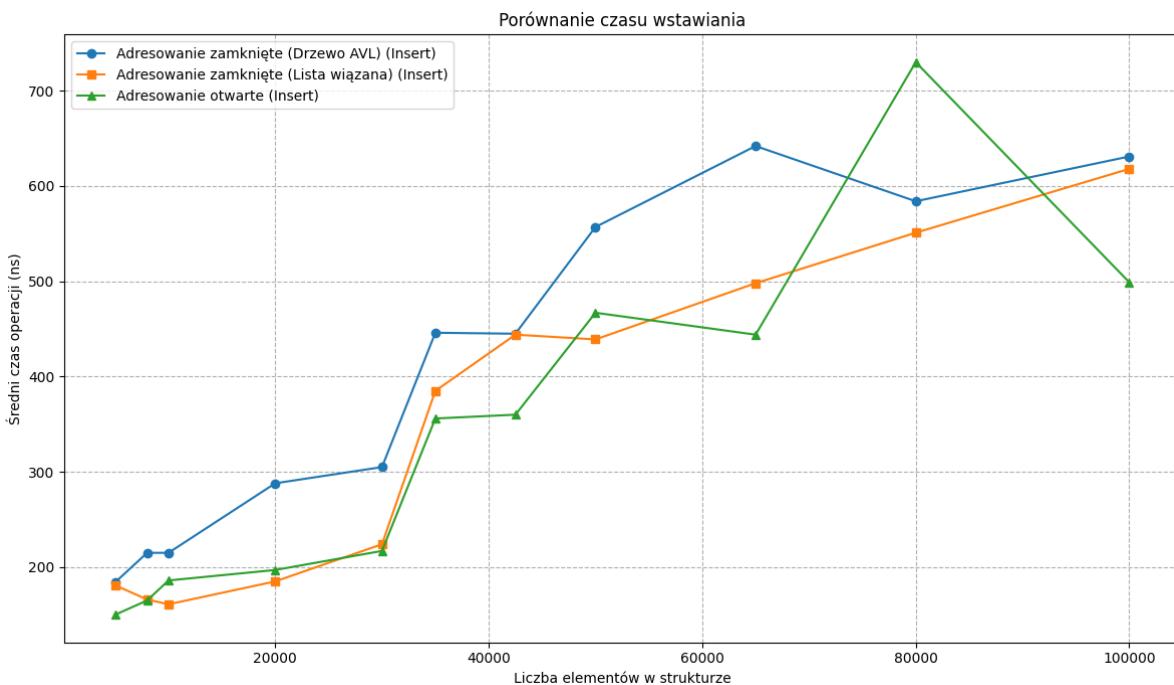
Rysunek 1: Przypadek średni dla tablicy mieszącej z adresowaniem otwartym



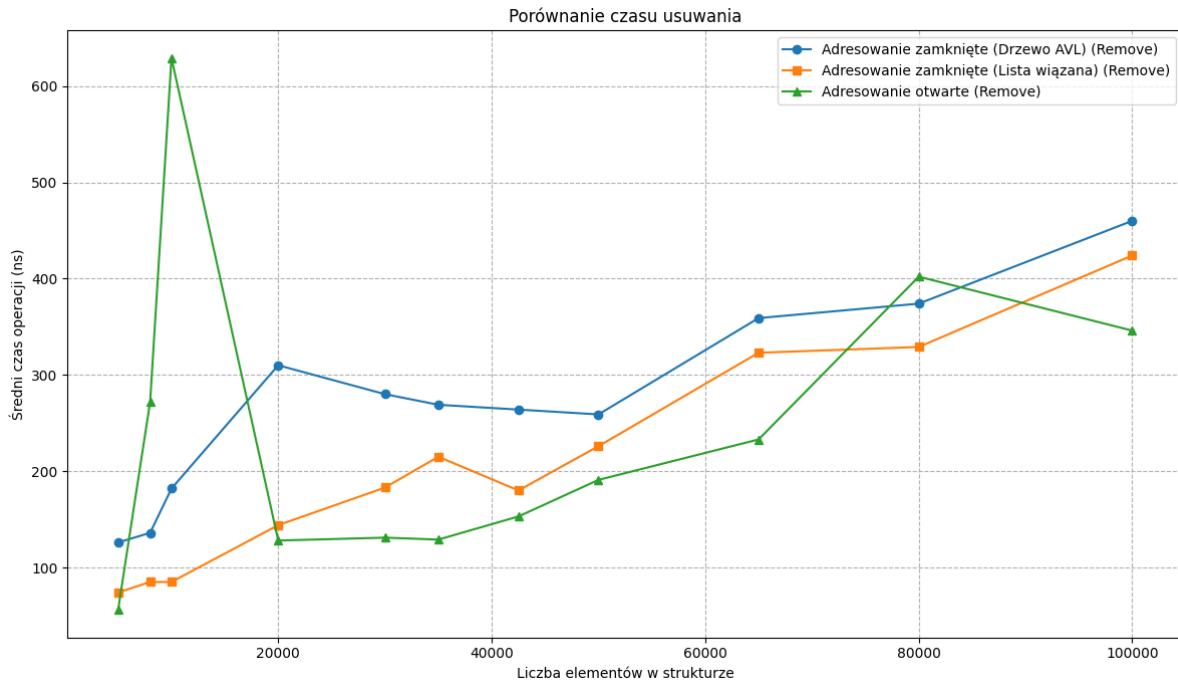
Rysunek 2: Przypadek średni dla tablicy mieszącej z adresowaniem zamkniętym opartym o listę wiązaną



Rysunek 3: Przypadek średni dla tablicy mieszającej z adresowaniem zamkniętym opartym o drzewo AVL



Rysunek 4: Porównanie dla operacji insert() dla przypadku średniego



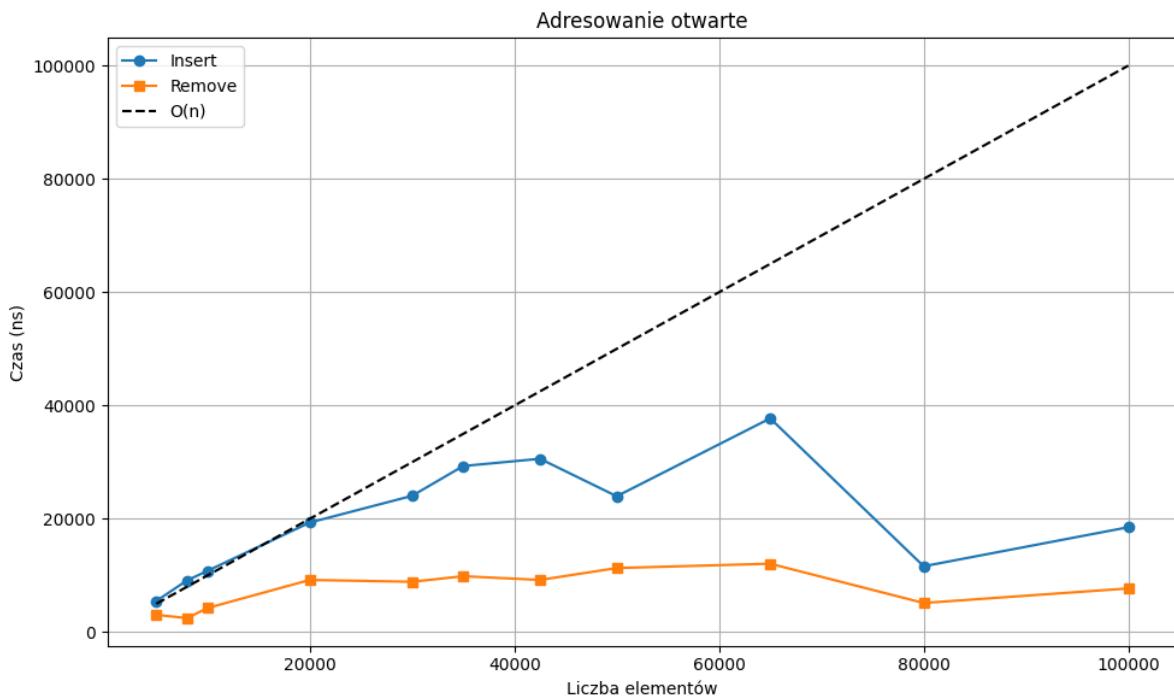
Rysunek 5: Porównanie dla operacji remove() dla przypadku średniego

### 3.2 Przypadek pesymistyczny

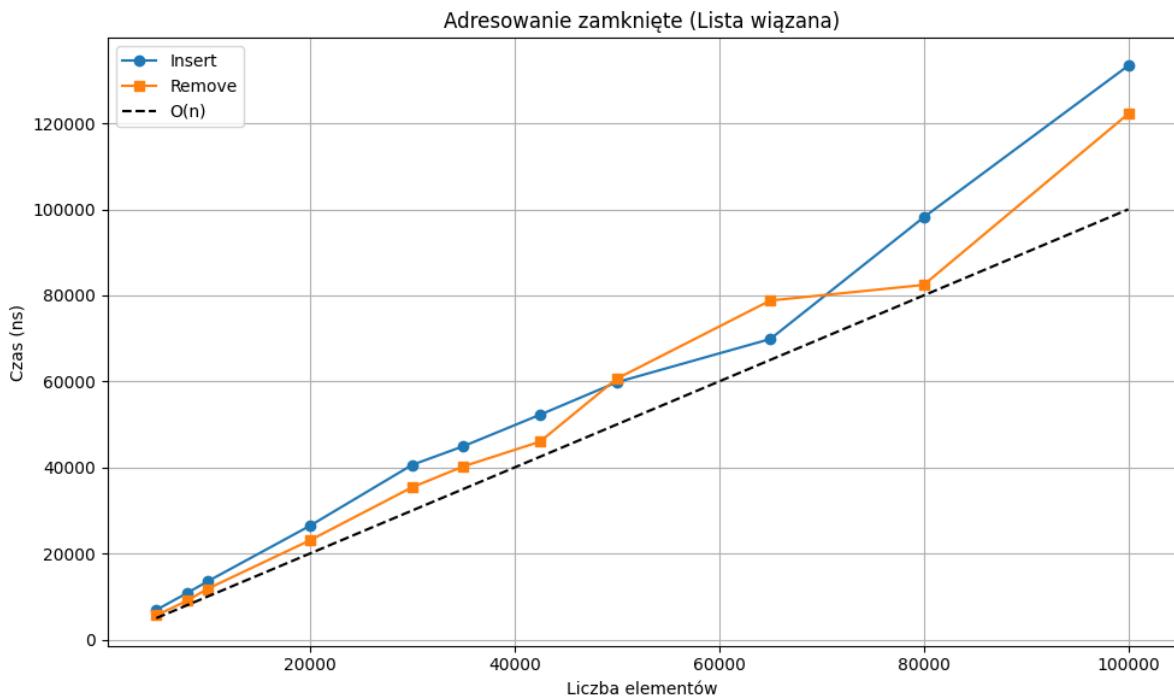
Przypadek pesymistyczny, maksymalizujemy ilość kolizji jaka może nastąpić, poprzez przypisywanie takich kluczy, które sprawiają iż funkcja hashująca będzie taka sama, ergo przypiszę wszystko do jednego kubelka/ na jeden indeks.

Tabela 4: Przypadek pesymistyczny dla wszystkich struktur

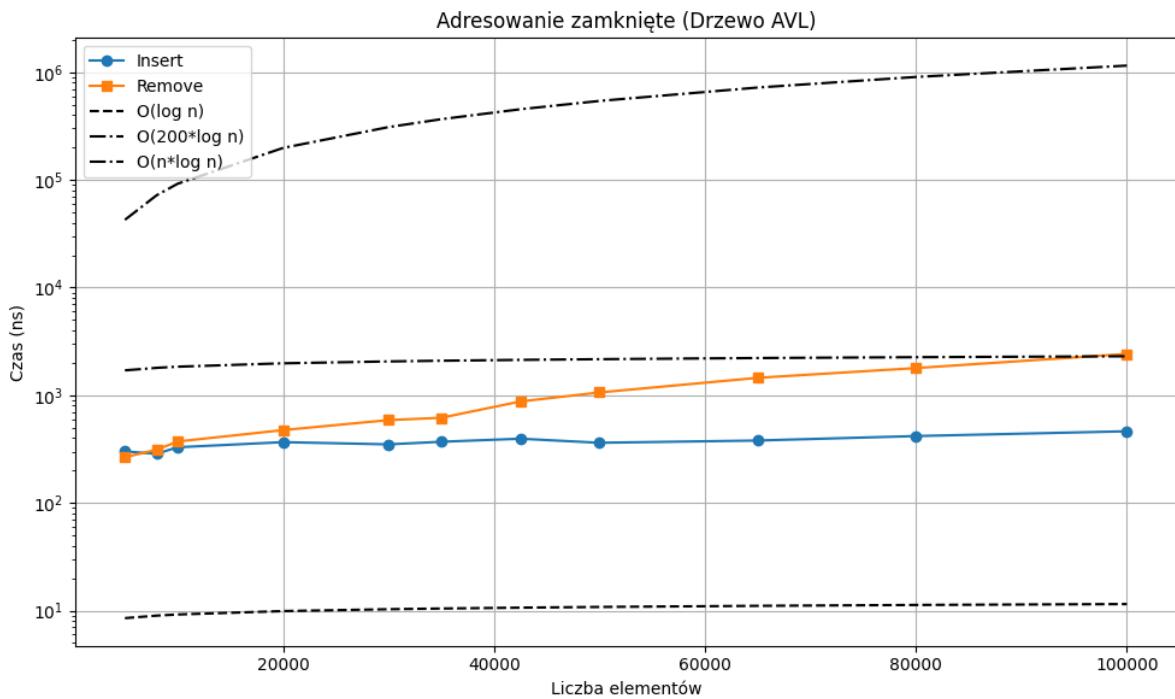
Liczba elementów	Adresowanie zamknięte (drzewo AVL)[ns]		Adresowanie zamknięte (lista wiązana)[ns]		Adresowanie otwarte [ns]	
	Wstawianie	Usuwanie	Wstawianie	Usuwanie	Wstawianie	Usuwanie
5 000	301	266	6 911	5 647	5 495	3 055
8 000	286	312	10 819	9 028	9 119	2 467
10 000	329	372	13 523	11 799	10 813	4 249
20 000	367	475	26 441	23 070	19 338	9 213
30 000	350	588	40 607	35 411	24 058	8 883
35 000	370	618	44 942	40 255	29 322	9 868
42 500	395	876	52 266	46 008	30 600	9 211
50 000	362	1 062	59 815	60 633	23 985	11 310
65 000	380	1 452	69 861	78 810	37 733	12 076
80 000	418	1 789	98 146	82 436	11 625	5 141
100 000	463	2 419	133 402	122 214	18 518	7 715



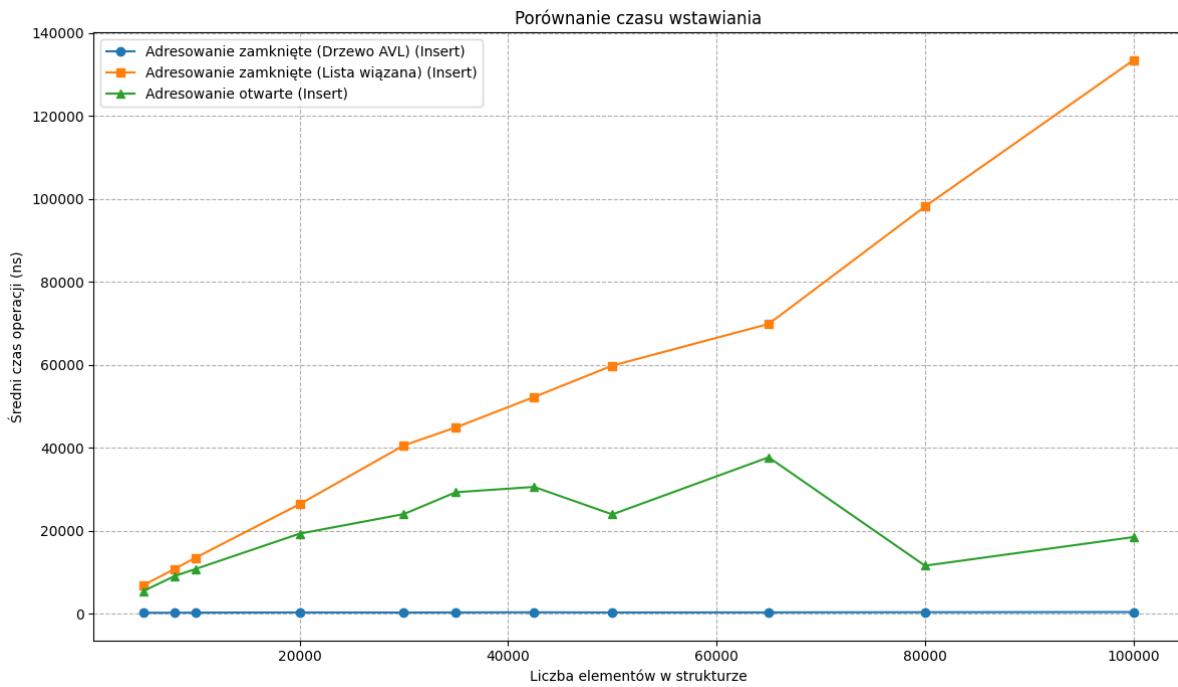
Rysunek 6: Przypadek pesymistyczny dla tablicy mieszącej z adresowaniem otwartym



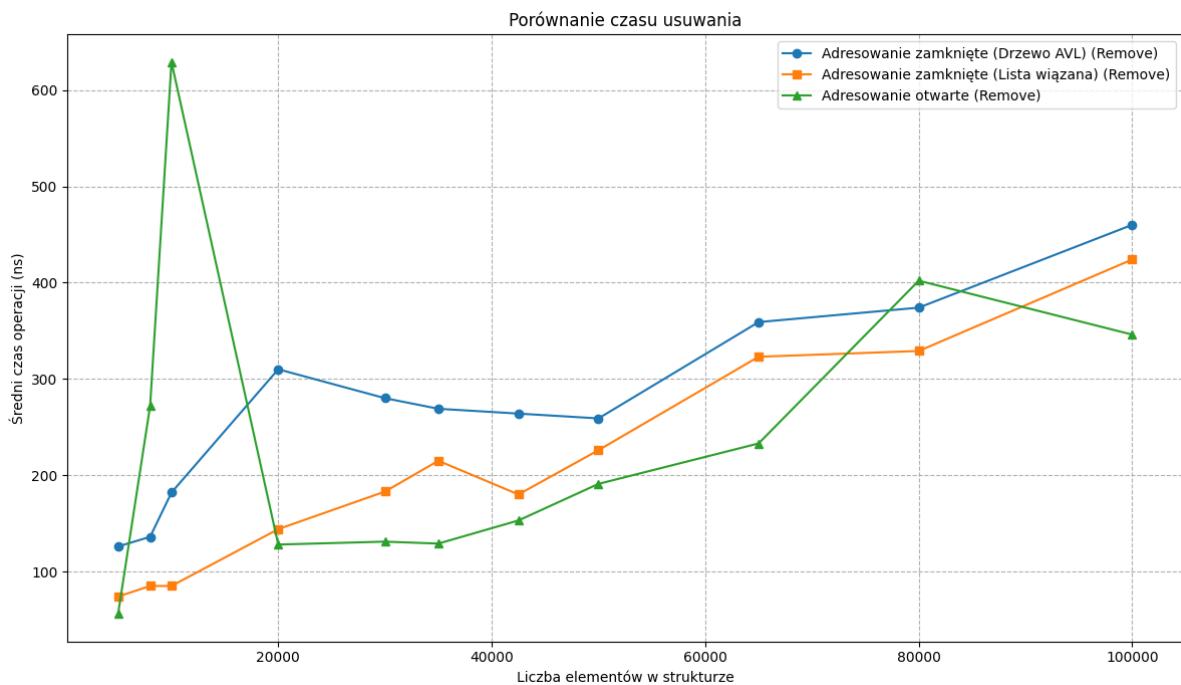
Rysunek 7: Przypadek pesymistyczny dla tablicy mieszącej z adresowaniem zamkniętym opartym o listę wiązaną



Rysunek 8: Przypadek pesymistyczny dla tablicy mieszającej z adresowaniem zamkniętym opartym o drzewo AVL



Rysunek 9: Porównanie dla operacji insert() dla przypadku pesymistycznego



Rysunek 10: Porównanie dla operacji remove() dla przypadku średniego

## 4 Wnioski

- Przypadek średni dla wszystkich struktur jest podobny do złożoności teoretycznej  $O(1)$  lub  $O(\log n)$ . Wariancje wyników możemy przypisać obciążeniu komputera lub innemu zjawisku losowemu.
- Przypadek pesymistyczny dla tablic mieszających z zamkniętym adresowaniem jest podobny do ich teoretycznej złożoności dla listy wiązanej  $O(n)$  a dla drzewa AVL  $O(\log n)$ .
- Adresowanie zamknięte oparte na drzewie AVL jest najszybsze dla pesymistycznego przypadku, co potwierdza złożoność teoretyczna. Natomiast adresowanie zamknięte oparte na liście wiązanej oraz adresowanie otwarte sprawują się lepiej dla przypadku średniego.
- Lista wiązana dla dobrze rozłożonej ilości danych jest szybsza od drzewa AVL, ponieważ wstawienie elementu na koniec lub usunięcie go dla listy dzieje się w czasie  $O(1)$ , natomiast drzewo AVL musi zostać zrównoważone.
- Dla pesymistycznego przypadku w adresowaniu otwartym jeśli generowanie kolidujących kluczy przekroczy pewną wartość, następuje zmiana klucza ze względu na generacje ich w następujący sposób:

$$Klucz = Klucz_{podstawowy} + (n * rozmiar)$$

Jeżeli dodawany element zdarzy się mieć klucz, który zmieści się w pozostawionej losowo dziurze, czas wykonania będzie niewielki, jest to szczególnie widoczne dla dużych liczb. Przy czym nadal pesymistyczna złożoność nadal będzie wynosić  $O(n)$ .

## 5 Źródła

Źródła wykorzystane do złożoności teoretycznej

- <https://www.bigocheatsheet.com>
- <https://kam.pwr.edu.pl/jaroslaw-rudypwr-edu-pl/files/sd/w8.pdf>