Struktury Danych	
Kierunek *Informatyczne Systemy Automatyki*	**Termin** *środa TN 15:15 – 16:55*
Imię, nazwisko, numer albumu *Jakub Wojtala 272542, Mateusz Marko 273168*	**Data** *10.04.2024r.*
Link do projektu https://github.com/kubikal7/Struktury-danych-1	

Sprawozdanie – miniprojekt nr 1

Struktury danych są fundamentalnym aspektem informatyki, umożliwiającym efektywne przechowywanie, organizację i przetwarzanie danych. W ramach tego projektu skupiamy się na implementacji i analizie trzech kluczowych struktur Tablicy Dynamicznej (ArrayList), Listy Jednokierunkowej (Singly Linked List) i Listy Dwukierunkowej (Doubly Linked List), ze szczególnym uwzględnieniem ich złożoności obliczeniowej w różnych scenariuszach operacyjnych. Jednym z kluczowych kryteriów oceny efektywności struktur jest złożoność obliczeniowa operacji na nich wykonywanych. U nas będzie to dodawanie, usuwanie i wyszukiwanie elementu.

Spis treści

1. Wstęp

- Wstęp teoretyczny

➢ **Tablica dynamiczna (ArrayList)** - struktura danych oparta na tablicy, która pozwala na dynamiczną zmianę rozmiaru w trakcie działania programu. Dzięki temu nie trzeba z góry określać maksymalnego rozmiaru struktury, co jest kluczową zaletą w porównaniu do statycznych tablic. Tablica dynamiczna automatycznie zwiększa swoją pojemność gdy istniejące już miejsce okazuje się zbyt małe. Jest szczególnie efektywna pod względem czasu dostępu do elementów, oferując stałą złożoność czasową *O(1)* dla operacji indeksowania.

- *Dodawanie elementu: O(1) w przypadku dodawania na koniec*, O(n) w innych przypakach, gdyż potrzebne jest przesunięcie elementów.
- *Usuwanie elementu:* O(1) w przypadku usuwania elementu z końca, *O(n) w innych przypadkach ze względu na konieczność przesunięcia elementów.*
- *Wyszukiwanie elementu: O(n)* w przypadku przeszukiwania liniowego.

➢ **Lista jednokierunkowa (Singly Linked List)** - składa się z serii węzłów, z których każdy przechowuje dane oraz wskaźnik do następnego węzła. Charakterystyczną cechą tej struktury jest to, że węzły nie muszą być przechowywane w sposób ciągły w pamięci. Pozwala to na elastyczne zarządzanie pamięcią i łatwe dodawanie lub usuwanie elementów bez konieczności przemieszczania pozostałych. Lista ta zapewnia szybki dostęp do pierwszego elementu. Jednakże dostęp do dalszych elementów wymaga przeglądania kolejnych węzłów, co może być czasochłonne dla dużych zbiorów.

- *Dodawanie elementu: O(1)* na początku, końcu (w przypadku struktury ze wskaźnikiem na koniec listy) i w środku listy, *O(n)* na końcu listy przy braku wskaźnika na ostatni element.
- *Usuwanie elementu: O(1)* przy usuwaniu głowy, *O(n)* dla pozostałych przypadków.
- *Wyszukiwanie elementu: O(n)* wymaga przeszukiwania kolejnych elementów.

➢ **Lista dwukierunkowa. (Doubly Linked List)** - rozszerza koncepcje listy jednokierunkowej o dodatkową referencję w każdym węźle, wskazującą na poprzedni element. Taka dwustronna nawigacja pozwala na efektywniejsze przeszukiwanie danych, zarówno od początku do końca listy, jak i na odwrót. Eliminuje to jedną z głównych wad list jednokierunkowych, umożliwiając szybsze usuwanie elementów oraz dodawanie nowych w dowolnym miejscu struktury. Mimo większego zużycia pamięci w porównaniu do list jednokierunkowych, listy dwukierunkowe oferują większą elastyczność i wydajność w wielu scenariuszach użycia.

- *Dodawanie elementu:* *O(1)* dla wszystkich przypadków.
- *Usuwanie elementu:* *O(1)* dla wszystkich przypadków.
- *Wyszukiwanie elementu:* *O(n)* podobnie jak w liście jednokierunkowej, możliwość natomiast tutaj wyszukiwania od końca.

• Założenia projektowe

➢ Celem projektu jest zbadanie złożoności czasowej podstawowych operacji na implementowanych strukturach danych. Eksperymenty obejmować będą operacje dodawania, usuwania i wyszukiwania elementów (w naszym przypadku na tablicy ze 120.000 elementami) na różnych pozycjach struktury (początek, koniec, wybrane miejsce). Generowanie elementów odbyło się poprzez wylosowanie liczb całkowitych z zakresu od 1 do 10 000 oraz zapisu tych liczb do pliku. Czas działania operacji zostanie zmierzony dla różnych rozmiarów struktur, z zakresami rozmiarów eksperymentalnie dobieranymi do wydajności sprzętu. Wyniki pomiarów przez nas tu podawanych będą średnią ze 100 prób.

➢ Eksperymenty przeprowadzone zostały na stacjonarnym komputerze wyposażonym w procesor Intel Core i-7-13700H, dysk SSD i 16 GB pamięci RAM. Oprogramowanie użyte do implementacji i testów to środowisko programistyczne Visual Studio, z wykorzystaniem języka C++ bez dostępu do gotowych bibliotek takich jak STL, co wymagało samodzielnej implementacji wszystkich struktur danych.

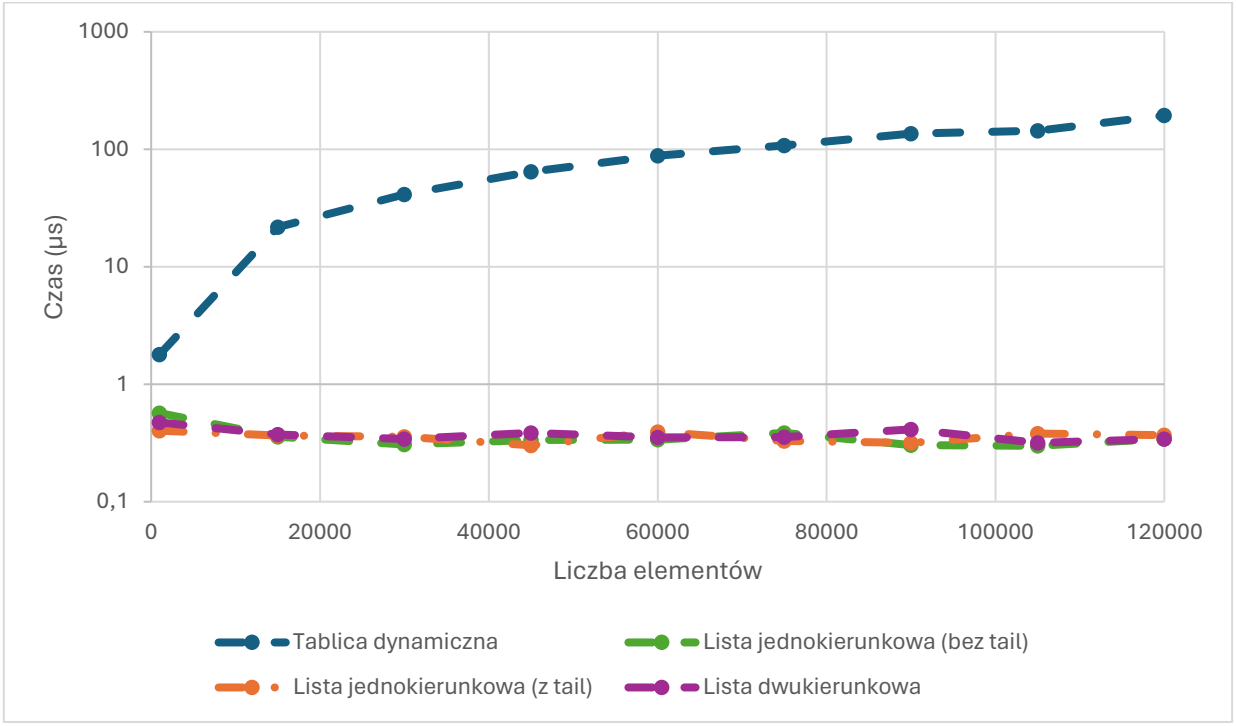
2. Badania

a) Operacja dodawania elementów do struktury

- *Dodawanie na początku*

Tabela 1 Dodawanie na początku - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	1,79	21,80	41,24	64,45	88,05	107,69	135,76	143,60	194,09
Lista jednokierunkowa (bez tail)	0,57	0,36	0,31	0,34	0,34	0,38	0,30	0,30	0,34
Lista jednokierunkowa (z tail)	0,40	0,37	0,36	0,30	0,39	0,33	0,32	0,38	0,37
Lista dwukierunkowa	0,47	0,37	0,34	0,39	0,35	0,35	0,41	0,32	0,34

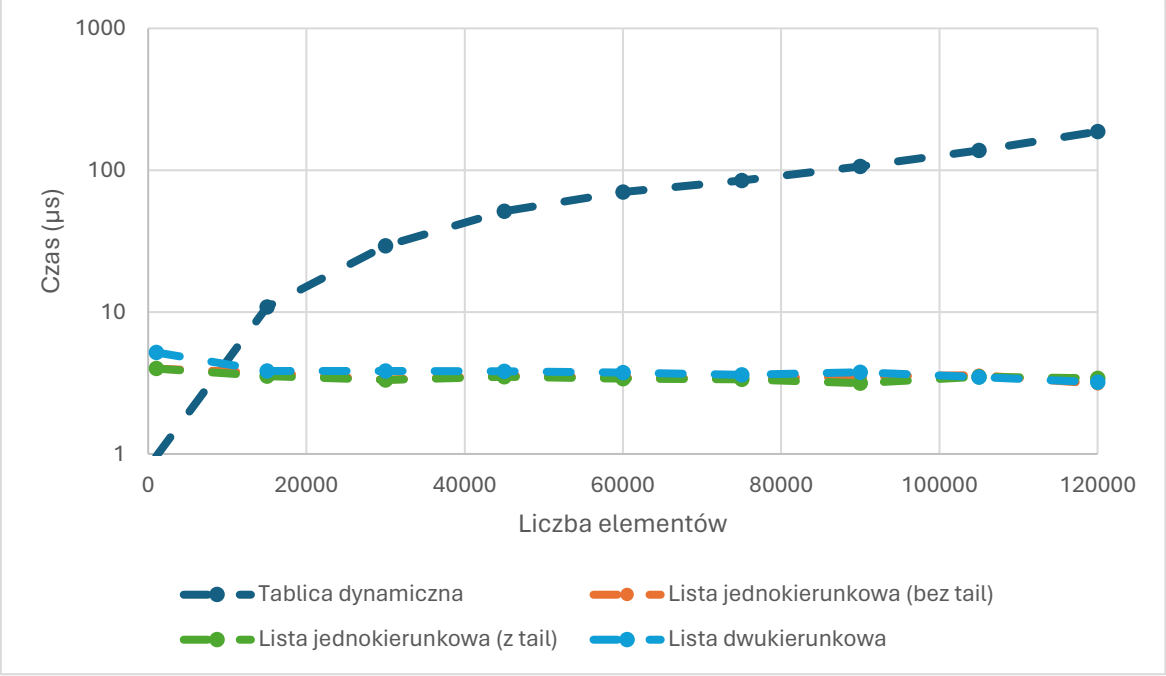


Rysunek 1 Wykres zależności czasu wykonania operacji dodawania elementów na początek od liczby elementów dla różnych struktur danych

- *Dodawanie na środkowy indeks*

Tabela 2 Dodawanie na środkowy indeks - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	0,96	10,87	29,37	51,52	70,36	84,43	106,21	137,82	187,15
Lista jednokierunkowa (bez tail)	3,99	3,73	3,33	3,57	3,44	3,41	3,51	3,59	3,12
Lista jednokierunkowa (z tail)	4,00	3,53	3,33	3,50	3,40	3,36	3,16	3,51	3,41
Lista dwukierunkowa	5,20	3,84	3,84	3,82	3,74	3,61	3,76	3,49	3,20

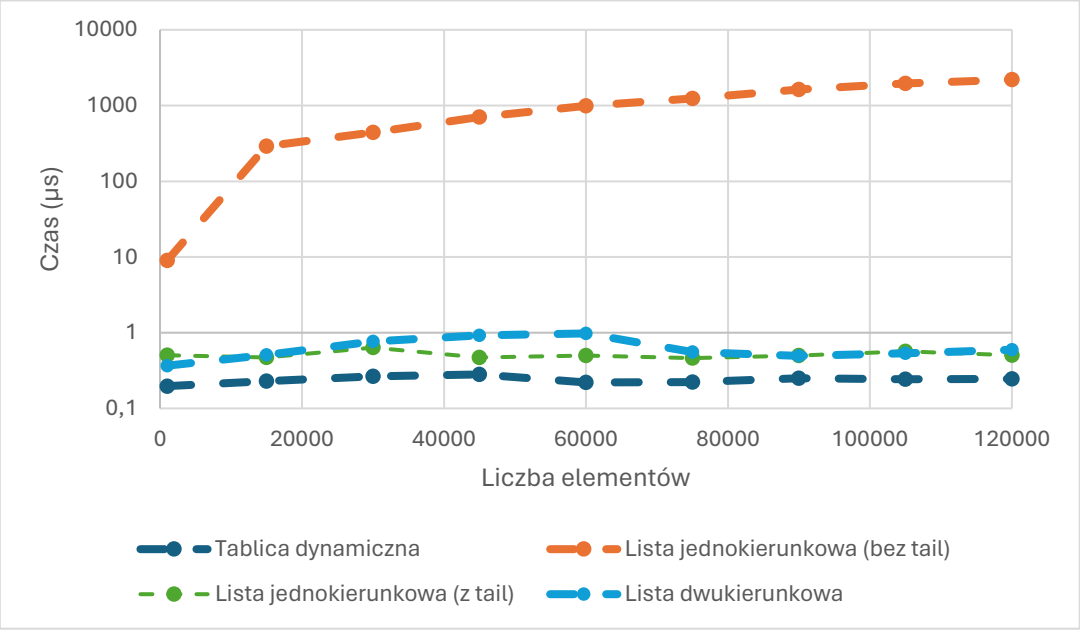


Rysunek 2 Wykres zależności czasu wykonania operacji dodawania elementów na środkowy indeks od liczby elementów dla różnych struktur danych

- *Dodawanie na koniec*

Tabela 3 Dodawanie na koniec - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	0,20	0,23	0,27	0,28	0,22	0,22	0,25	0,24	0,25
Lista jednokierunkowa (bez tail)	8,97	291,58	442,85	703,93	986,98	1234,13	1630,90	1950,80	2193,98
Lista jednokierunkowa (z tail)	0,50	0,47	0,64	0,47	0,50	0,46	0,50	0,57	0,50
Lista dwukierunkowa	0,37	0,51	0,77	0,92	0,98	0,56	0,49	0,53	0,60



Rysunek 3 Wykres zależności czasu wykonania operacji dodawania elementów na koniec od liczby elementów dla różnych struktur danych

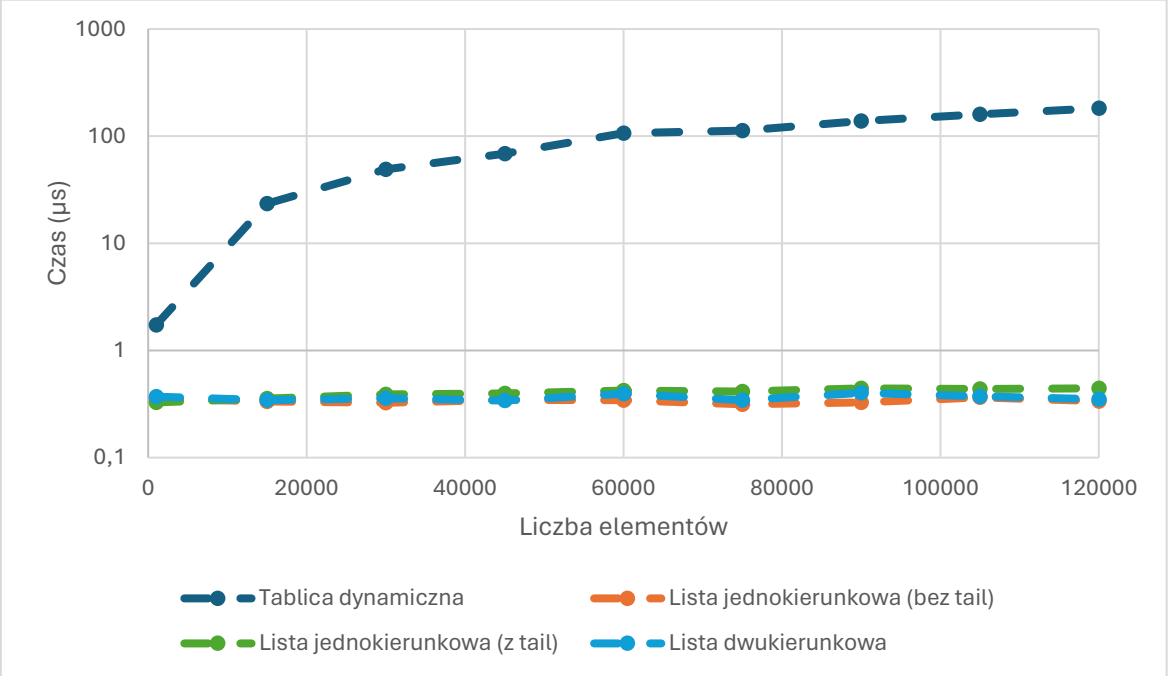
Możemy zauważyć, że dodawanie na początek było szybsze dla list, gdyż operacja ta wymaga tylko zmiany wskaźnika head. W tablicy dynamicznej znacznie wolniej, gdyż trzeba przesunąć wszystkie elementy. Dodawanie na końcu z kolei nie jest efektywne dla listy jednokierunkowej bez tail, ponieważ wymagane jest przejście każdego elementu.

b) Operacja usuwania elementów ze struktury

- *Usuwanie z początku*

Tabela 4 Usuwanie na początku - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	1,73	23,54	49,00	68,67	106,79	112,61	138,61	159,82	182,13
Lista jednokierunkowa (bez tail)	0,36	0,33	0,33	0,35	0,35	0,32	0,33	0,37	0,34
Lista jednokierunkowa (z tail)	0,33	0,36	0,39	0,40	0,42	0,42	0,44	0,44	0,44
Lista dwukierunkowa	0,37	0,34	0,36	0,34	0,40	0,35	0,40	0,37	0,35

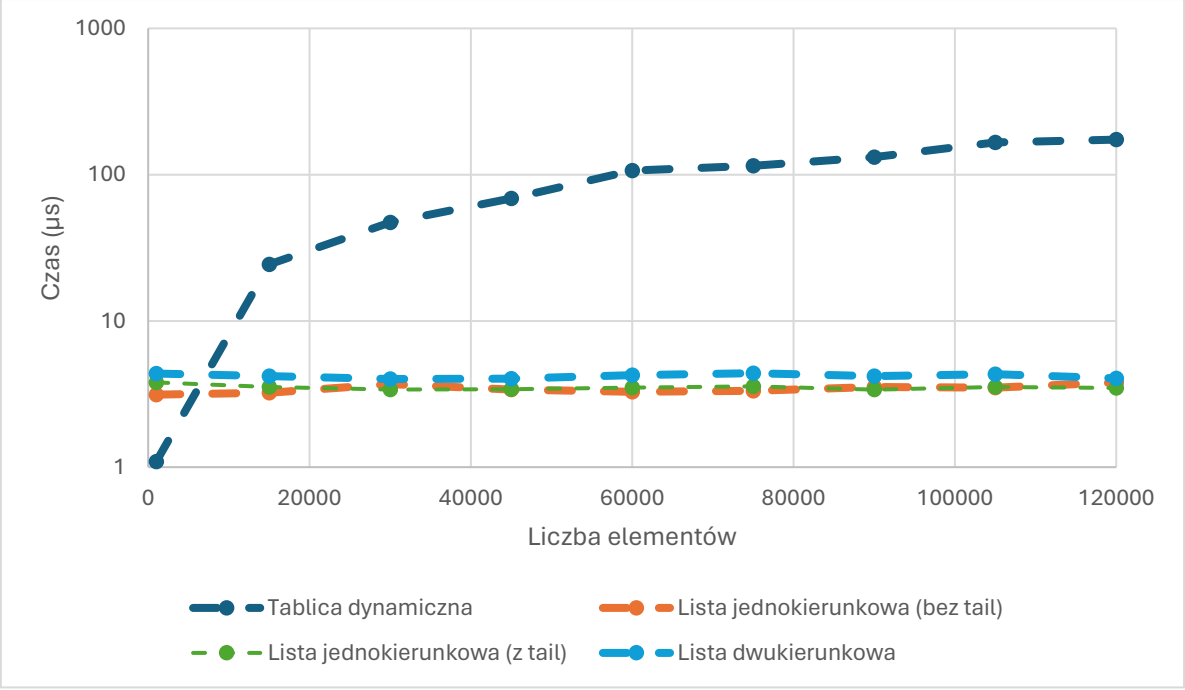


Rysunek 4 Wykres zależności czasu wykonania operacji usuwania elementów z początku od liczby elementów dla różnych struktur danych

- *Usuwanie ze środka*

Tabela 5 Usuwanie ze środka - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	1,09	24,38	47,17	68,78	106,40	114,98	131,96	165,98	173,51
Lista jednokierunkowa (bez tail)	3,13	3,23	3,69	3,39	3,28	3,33	3,54	3,49	3,79
Lista jednokierunkowa (z tail)	3,80	3,54	3,40	3,41	3,49	3,57	3,39	3,53	3,48
Lista dwukierunkowa	4,37	4,19	4,01	4,02	4,26	4,39	4,19	4,31	4,06

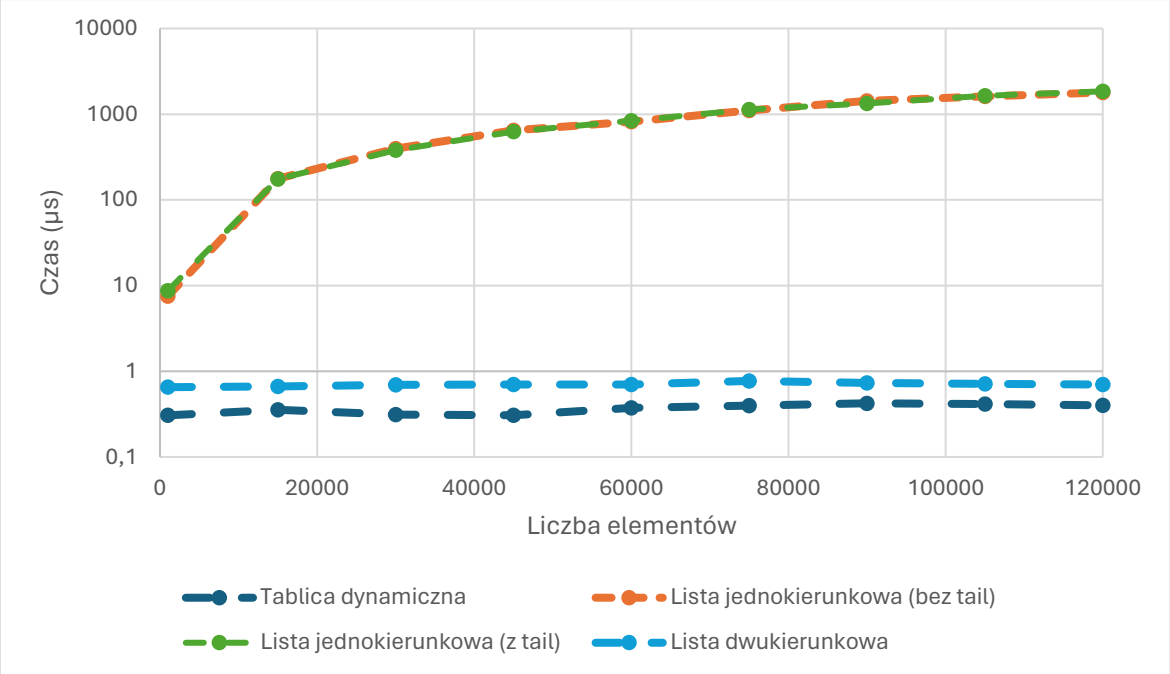


Rysunek 5 Wykres zależności czasu wykonania operacji usuwania elementów ze środka od liczby elementów dla różnych struktur danych

- *Usuwanie z końca*

Tabela 6 Usuwanie z końca - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury

	Liczba elementów								
	1000	15 000	30 000	45 000	60 000	75 000	90 000	105 000	120 000
Tablica dynamiczna	0,31	0,36	0,31	0,31	0,38	0,40	0,42	0,42	0,40
Lista jednokierunkowa (bez tail)	7,54	177,20	398,30	650,35	818,47	1101,73	1425,12	1606,29	1789,62
Lista jednokierunkowa (z tail)	8,71	175,42	380,14	627,24	841,40	1133,43	1332,18	1632,91	1845,08
Lista dwukierunkowa	0,65	0,67	0,69	0,70	0,70	0,77	0,73	0,71	0,70



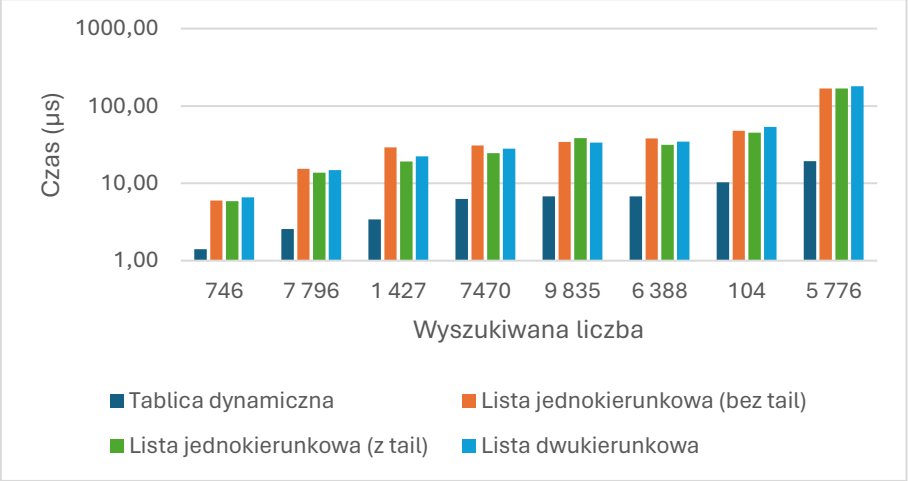
Rysunek 6 Wykres zależności czasu wykonania operacji usuwania elementów z końca od liczby elementów dla różnych struktur danych

Usunięcie elementu z przodu jest szybkie w listach, bo wymaga jedynie zmiany wskaźnika head, natomiast w tablicy dynamicznej musimy przesunąć wszystkie elementy. Listy dwukierunkowe pozwalają z kolei na szybkie usunięcie elementu z końca dzięki bezpośredniemu dostępowi do tail oraz tail->previous (przedostatni element przed usunięciem). W listach jednokierunkowych operacja ta jest jednak wolniejsza, gdyż trzeba przejść przez całą listę.

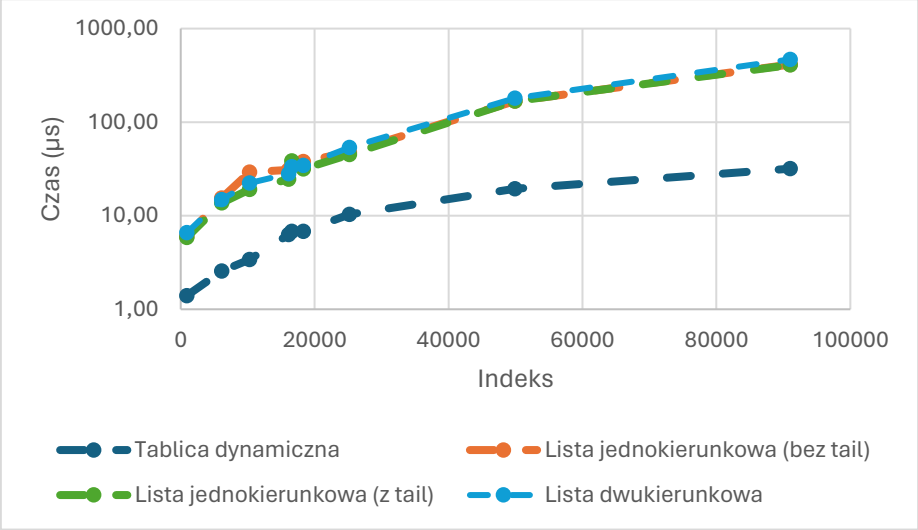
c) Operacja wyszukiwania elementu w strukturze

Tabela 7 Wyszukiwanie danego elementu - czas operacji (µs) dla poszczególnej ilości elementów i rodzaju struktury wraz z indeksem

	Wyszukiwana liczba								
	746	7 796	1 427	7470	9 835	6 388	104	5 776	5 429
Index ->	(911)	(6115)	(10288)	(16115)	(16572)	(18323)	(25185)	(49905)	(91021)
Tablica dynamiczna	1,40	2,56	3,40	6,28	6,78	6,81	10,30	19,34	31,80
Lista jednokierunkowa (bez tail)	6,00	15,40	29,18	30,99	34,17	37,87	47,76	168,01	412,55
Lista jednokierunkowa (z tail)	5,87	13,74	19,08	24,51	38,58	31,55	45,15	167,73	407,83
Lista dwukierunkowa	6,57	14,88	22,30	28,12	33,52	34,52	53,49	179,79	468,22



Rysunek 7 Wykres słupkowy przedstawiający zestawienie czasów wykonania operacji wyszukiwania konkretnego elementu dla różnych struktur



Rysunek 8 Wykres zależności czasu wykonania operacji wyszukania elementu dla różnych struktur danych

Politechnika Wrocławska

Wyszukiwanie w listach jedno i dwukierunkowych jest wolne, gdyż wymaga przeglądania wszystkich elementów po kolei (złożoność *O(n)*). W tablicy dynamicznej choć teoretycznie też jest to wyszukiwanie liniowe, dostęp do elementów jest szybszy z uwagi na ciągłą pamięć i lepsze wykorzystanie cache procesora. Czas wyszukiwania może się jednak różnić w zależności od położenia szukanego elementu.

3. Wnioski

- Tablica dynamiczna jest najlepsza dla operacji indeksowania i dostępu do elementów z czasem dostępu O(1). Idealna dla scenariuszy wymagających szybkiego dostępu do danych i nieznanej z góry liczby elementów. Jednakże Dodawanie i usuwanie elementów na początku i w środku jest nieefektywne z powodu konieczności przesuwania elementów.
- Lista jednokierunkowa sprawdza się w operacjach dodawania i usuwania na początku listy. Dobry wybór dla aplikacji, gdzie operacje dodawania i usuwania są częstsze niż odczyty i nie ma potrzeby szybkiego dostępu do losowych elementów. Ma problem z operowaniem na ostatnich elementach (szczególnie lista bez wskaźnika na ostatni element), a jej sekwencyjność sprawia, że wyszukiwanie i dostęp do elementów jest wolniejszy w porównaniu do tablic dynamicznych.
- Lista dwukierunkowa zapewnia z kolei lepszą elastyczność niż lista jednokierunkowa, umożliwiając efektywne operacje na obu końcach listy. Sprawdzi się dla sytuacji, które wymagają częstego dodawania i usuwania elementów nie tylko na początku, ale i na końcu listy. Jednak ona także posiada ograniczenia. Zużycie pamięci będzie większe przez dodatkowe wskaźniki na poprzednie elementy co może mieć wpływ przy dużych ilościach danych.
- Dla małych i średnich zbiorów danych różnice w wydajności mogą być małe, jednak zwiększając liczbę elementów widzimy, że się to zmienia. Dla dużych zbiorów efektywność operacji ogółem spada. Na listach jednokierunkowych i dwukierunkowych przy operacjach wymagających dostępu do losowych elementów widać mniejszą skuteczność, natomiast tablice dynamiczne utrzymują dobre czasy.
- Podsumowując wybór struktury danych powinien być podyktowany konkretnymi wymaganiami, takimi jak częstość i typ operacji (dodawanie, usuwanie, wyszukanie), oraz przewidywany rozmiar danych.

4. Bibliografia

- http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/classes.php?lang=pl#
- https://www.geeksforgeeks.org/data-structures/linked-list/
- https://www.javatpoint.com/ds-array-vs-linked-list
- https://www.prepbytes.com/blog/arrays/data-structures-in-c-array-linked-list-stack/