Spis treści

[Cel projektu 1](#_Toc162778036)

[Wybrany język 1](#_Toc162778037)

[Tablica wbudowana 1](#_Toc162778038)

[Tablica jednokierunkowa 2](#_Toc162778039)

[Tablica dwukierunkowa 2](#_Toc162778040)

[Testowanie 3](#_Toc162778041)

[Wyniki Testów 3](#_Toc162778042)

[Podsumowanie 3](#_Toc162778043)

[Odwołania 4](#_Toc162778044)

# Cel projektu

Projekt polega na porównaniu czasu wykonywania operacji usuwania, dodawania, wyszukiwania na początku, końcu i środku na różnych typach tablic. Takich jak Tablica domyślnie zadeklarowana w wybranym języku, Tablica jednokierunkowa jak i Tablica dwukierunkowa które zostaną na potrzeby projektu napisane od podstaw. Dane testowania będą wygenerowane przez prosty kod w javie i będą to dodatnie liczby całkowite w zakresie od 0-100000 w ilości 100000.

# Wybrany język

Język wybrany do tego projektu to Java która umożliwi nam spełnienie projektu używając obiektowości.

# Tablica wbudowana

Wybraliśmy klasę ArrayList która ma od razu zaimplementowane zmianę rozmiaru w przypadku dodawania/usuwania elementów. ArrayList pozwala na przechowywanie wszystkich typów nawet null’ów. [1]

Dodawania:

* na końcu O(1) [2]

Wyszukiwanie:

* O(n) [2]

Usuwanie:

* O(n) [2]

# Tablica jednokierunkowa

Tablica jednokierunkowa to struktura danych gdzie każda wartość jest osobnym obiektem który jest nazwany węzłem (node). Każdy węzeł zawiera dane takie jak przechowywaną wartość, czy jest początkiem łańcucha i informacje gdzie znajduje się następny element. [3]

Złożoność obliczeniowa dla operacji:

Dodawania:

* na początku O(1) [4]
* w środku O(n) [4]
* na końcu O(1) [4]

Wyszukiwanie:

* O(n) [4]

Usuwanie:

* na początku O(1) [4]
* w środku O(n) [4]
* na końcu O(1) [4]

# Tablica dwukierunkowa

Tablica dwukierunkowa to struktura danych która zawiera węzły, węzły mają swoje elementy takie jak: głowa (head) początek węzłów, gdzie znajduje się następny i poprzedni węzeł. [5]

Złożoność obliczeniowa dla poszczególnych operacji wygląda następująco:

Dodawania:

* na początku O(1) [4]
* w środku O(n) [4]
* na końcu O(1) [4]

Wyszukiwanie:

* O(n) [4]

Usuwanie:

* na początku O(1) [4]
* w środku O(n) [4]
* na końcu O(1) [4]

# Testowanie

Pod testowanie zaliczamy czas wykonania poszczególnych operacji, wstawiania, usuwania, wyszukiwania na tablicach: dwukierunkowej, jednokierunkowej, i wbudowanej w programie.  
Czas będzie liczony w milisekundach.

# Wyniki Testów

Obraz zawierający tekst, Czcionka, linia, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

# Podsumowanie

Zacznijmy od analizowania wstawiania jak możemy zauważyć wstawianie w liście jednokierunkowej i dwukierunkowej jest wolne, ale lista dwukierunkowa jest wolniejsza od wszystkich, może to wynikać z zmienianiem wartości prieviousNode i nextNode, gdy lista jednokierunkowa ma tylko nextNode. Żeby dodać trzeba też najpierw „znaleźć” koniec tablicy za każdym razem gdy chcemy coś dodać. Tablica wbudowana która została użyta w programie ma najniższy czas ponieważ zawsze dodaje na koniec tablicy.

Drugi słupek odpowiada za wyszukiwanie w tablicy. Patrząc na złożoność obliczeniową listy jednokierunkowej i dwukierunkowej można zauważyć że lista dwukierunkowa wypadła dużo lepiej od pierwszej. Jest to spowodowane tym że: Lista jednokierunkowa jest oszczędzaczem pamięci a lista dwukierunkowa jest leprze do wyszukiwania ponieważ można się cofać pomiędzy węzłami. [6] Prędkość w tablicy wbudowanej jest mniejsza zależy to od struktury przechowywania danych w takiej tablicy.

Trzeci słupek czyli usuwanie pokazuje że usuwanie w liście jednokierunkowej i dwukierunkowej jest najszybsze ponieważ usuwając element usuwamy węzły poprzez odłączenie połączeń pomiędzy nimi, język Java ma wbudowany „Garbage Collector” i gdy widzi węzeł do którego nie da się odnieść w żaden sposób odnieść po prostu go usuwa więc nie musimy się aż tak martwić usuwaniem. ArrayList jednak musi dokonać zmiany indexów w swoich danych przy usuwaniu dlatego tutaj ArrayList jest najwolniejszy.

# Odwołania

|  |  |
| --- | --- |
| [1] | O. (ArrayList), „docs.oracle.com,” oracle, [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html. |
| [2] | K. Ślusarz, „kompikownia.p,” Kompikownia, [Online]. Available: https://www.kompikownia.pl/index.php/2021/04/22/java-internals-jak-dziala-arraylist-jezyku-java/. |
| [3] | S. J. (SLL), „javatpoint.com,” javatpoint, [Online]. Available: https://www.javatpoint.com/java-program-to-create-and-display-a-singly-linked-list. |
| [4] | g. (. S. DLL), „geeksforgeeks.com,” geeksforgeeks, [Online]. Available: https://www.geeksforgeeks.org/time-and-space-complexity-of-linked-list/. |
| [5] | g. (DLL), „geeksforgeeks.org,” geeksforgeeks, [Online]. Available: https://www.geeksforgeeks.org/data-structures/linked-list/doubly-linked-list/. |
| [6] | g. (. v. DLL), „geeksforgeeks.org,” geeksforgeeks, [Online]. Available: https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/. |