Struktury Danych i Złożoność Obliczeniowa

Zadanie projektowe 1

Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych

Marcin Wojciechowski
235088

Spis treści

1 Opis zadania projektowego

1.1 Cel zadania

Celem tego zadania jest implementacja następujących struktur danych:

- tablicy,
- listy,
- kopca binarnego,
- drzewa czerwono-czarnego,

oraz dokonanie pomiaru czasu działania operacji dodawania, usuwania oraz wyszukiwania elementu w zaimplementowanych strukturach.

1.2 Wymogi implementacji

- Podstawowym elementem struktur jest 4 bajtowa liczba całkowita
- Wszystkie struktury powinny być alokowane dynamicznie
- Pomiary należy powtórzyć kilkukrotnie, a wynik uśrednić
- Dla tablicy i listy należy rozpatrzeć przypadki dodawania i usuwania elementu w zależności od pozycji w danej strukturze (początek, środek, koniec)

2 Złożoności obliczeniowe operacji

2.1 Opis złożoności obliczeniowej

Złożoność obliczeniowa jest definiowana jako określenie ilości zasobów potrzebnych do rozwiązania problemów obliczeniowych. Wyrażana jest jako funkcja parametru, od którego zależna jest jej wartość. Wyróżniamy dwa podstawowe typy złożoności obliczeniowej:

- Złożoność pamięciowa, czyli ilość pamięci która zostanie wykorzystana w celu realizacji danego algorytmu, bądź przechowania określonej ilości danych.
- Złożoność czasowa, czyli czas potrzebny na wykonanie danego algorytmu. Wyrażony zwykle w jednostkach czasu lub liczbie cykli procesora.

Dla algorytmów, które zależne są od zbioru danych, możemy wyróżnić trzy typy złożoności:

- Złożoność optymistyczna, w którym przyjmujemy najkorzystniejszy zbiór danych
- Złożoność oczekiwana (losowy zbiór danych)
- Złożoność pesymistyczna, w którym przyjmujemy najmniej korzystny zbiór danych.

2.2 Złożoności operacji dla poszczególnych struktur danych

2.2.1 Tablica

Funkcja	Złożoność oczekiwana	Złożoność pesymistyczna
Dodanie wartości	O(n)	O(n)
Usunięcie wartości	O(n)	O(n)
Wyszukanie wartości	O(n)	O(n)

2.2.2 Lista

Funkcja	Złożoność oczekiwana	Złożoność pesymistyczna
Dodanie wartości	O(-)	O(n)
Usunięcie wartości	O(-)	O(n)
Wyszukanie wartości	O(n)	O(n)

2.2.3 Kopiec binarny

Funkcja	Złożoność oczekiwana	Złożoność pesymistyczna
Dodanie wartości	O(1)	O(1)
Usunięcie wartości	O(1)	O(1)
Wyszukanie wartości	O(n)	O(n)

2.2.4 Drzewo czerwono-czarne

Funkcja	Złożoność oczekiwana	Złożoność pesymistyczna
Dodanie wartości	O(log(n))	O(log(n))
Usunięcie wartości	O(log(n))	O(log(n))
Wyszukanie wartości	O(log(n))	O(log(n))

3 Plan eksperymentu

3.1 Pomiar czasu

Do pomiaru czasu wykorzystana została biblioteka <time.h>, zawarta w zbiorze standardowych bibliotek języka C++. Biblioteka ta operuje na liczbie cykli procesora, co pozwala na bardzo dokładny pomiar czasu.

Program pobiera czas przed i po wykonaniu funkcji, a następnie oblicza różnicę czasu bazując na ilości okresów przebiegu procesora.

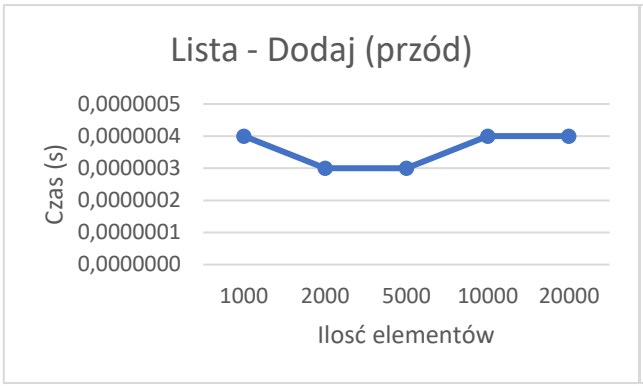
3.2 Przebieg pomiarów

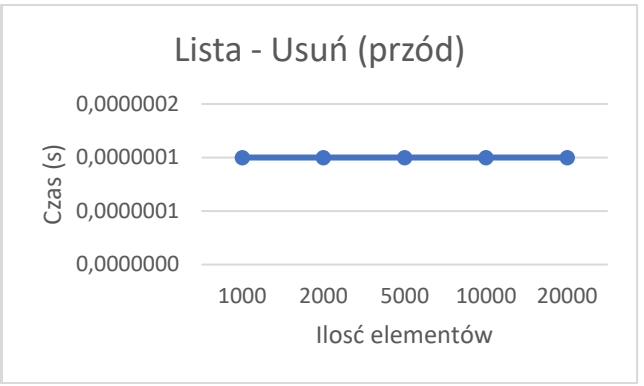
Będziemy mierzyć czas wykonywania operacji w funkcji rozmiaru struktury (liczymy średni czas wykonania pojedynczej operacji). Jako że wynik zależy od rozkładu danych (generacja liczb

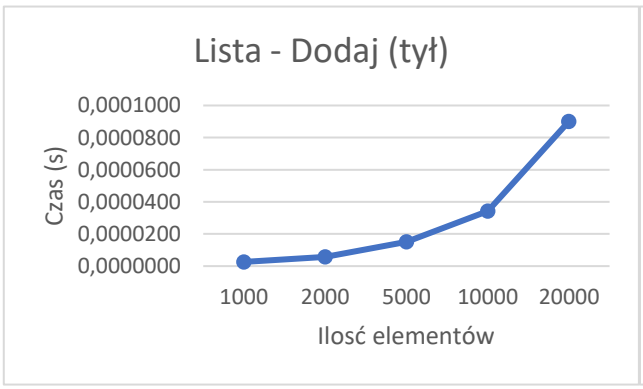
pseudolosowych) pomiar zostanie powtórzony kilkudziesięciokrotnie, a następnie zostanie uśredniony.

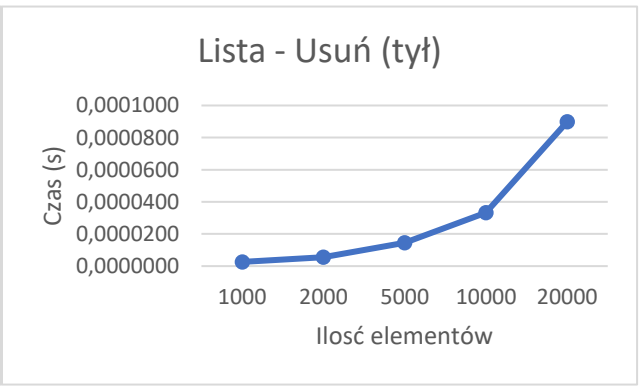
4 Zestawienie wyników

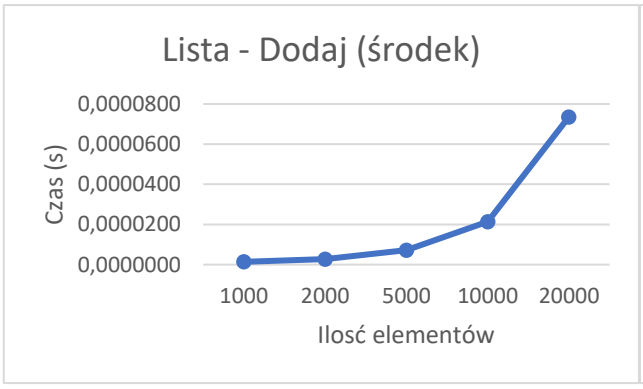
4.1 Lista

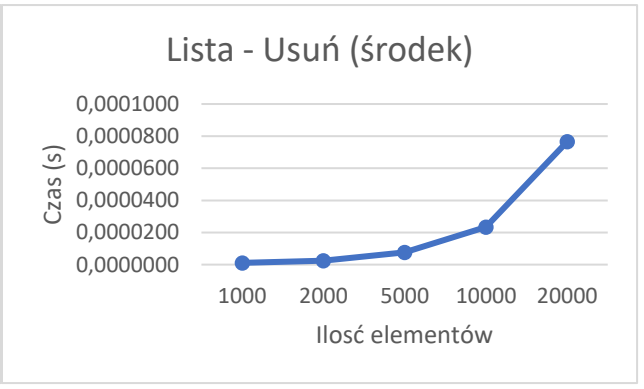


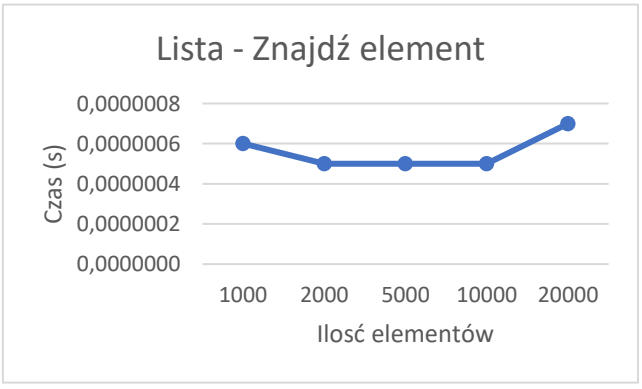




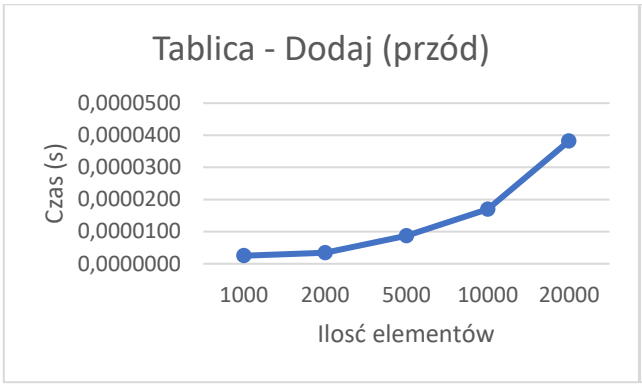


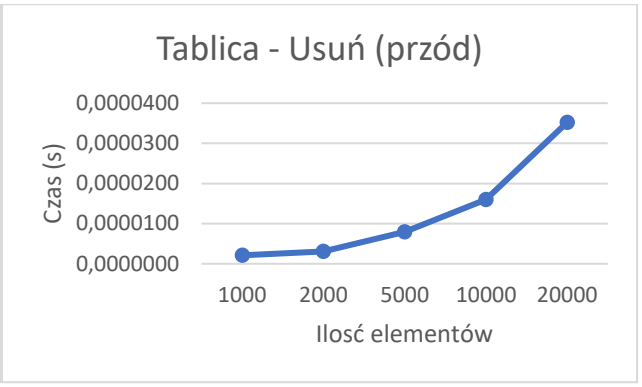


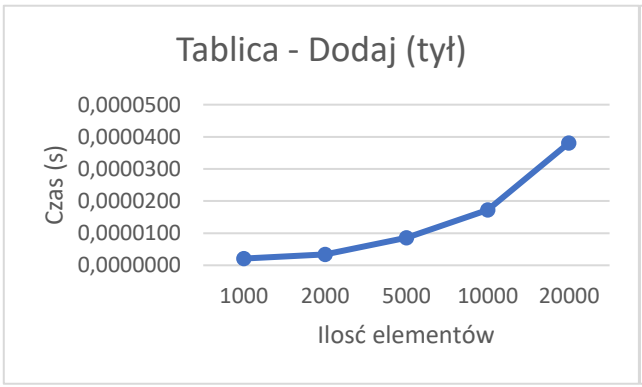


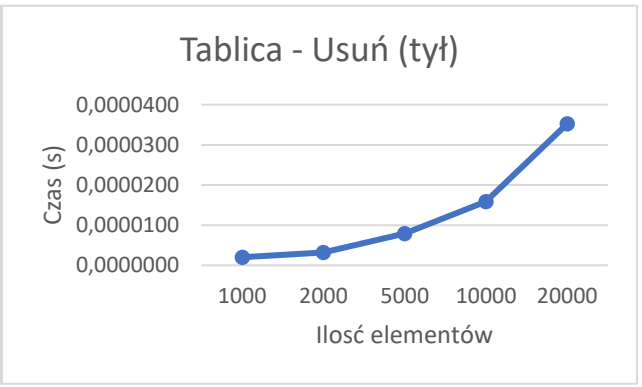


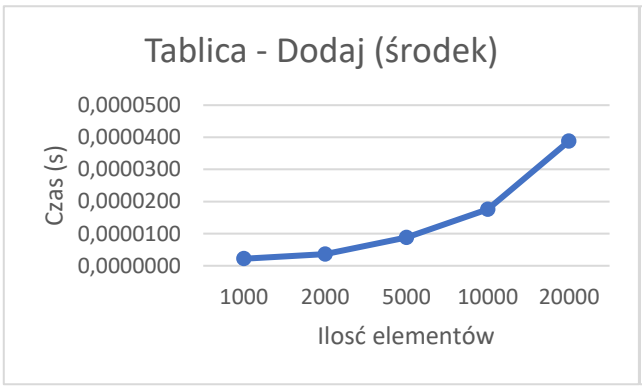
4.2 Tablica

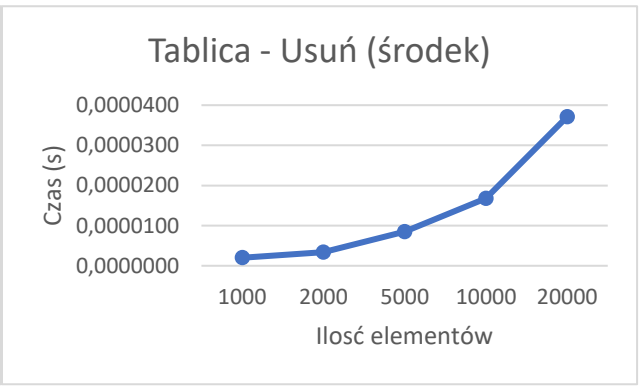

Tablica - Dodaj (przód)
Czas (s)
0,0000500
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów

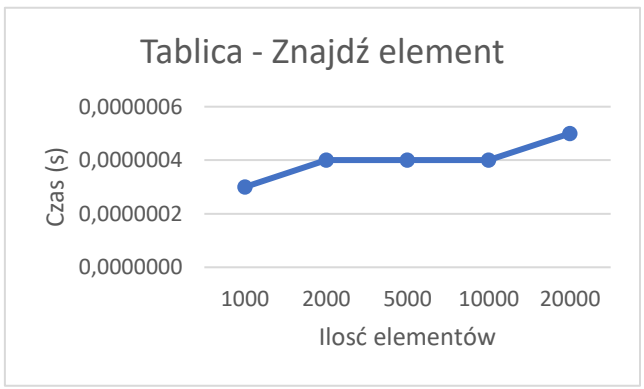

Tablica - Usuń (przód)
Czas (s)
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów


Tablica - Dodaj (tył)
Czas (s)
0,0000500
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów

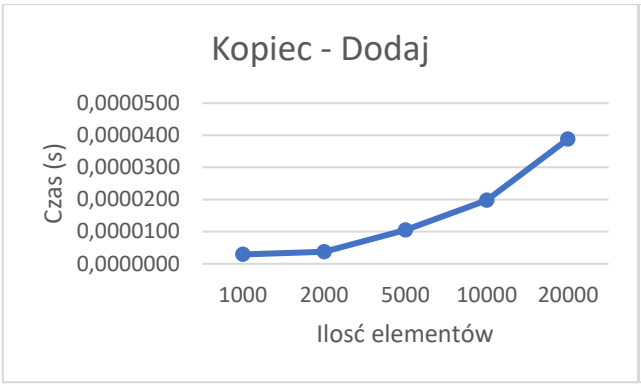

Tablica - Usuń (tył)
Czas (s)
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów

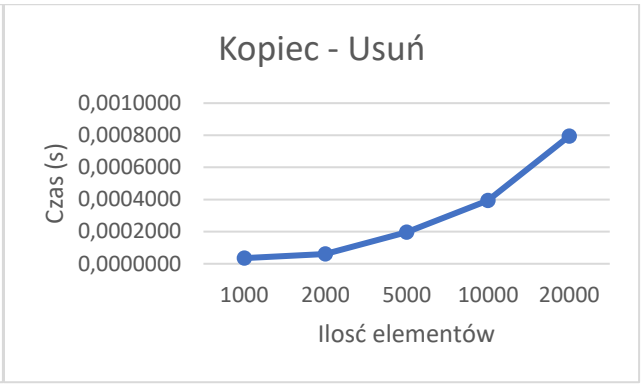

Tablica - Dodaj (środek)
Czas (s)
0,0000500
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów

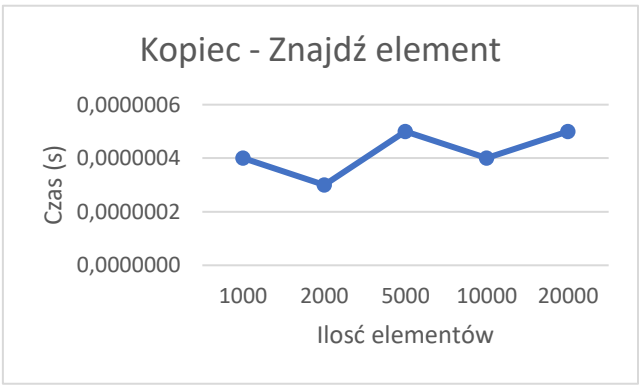

Tablica - Usuń (środek)
Czas (s)
0,0000400
0,0000300
0,0000200
0,0000100
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów


Tablica - Znajdź element
Czas (s)
0,0000006
0,0000004
0,0000002
0,0000000
1000
2000
5000
10000
20000
Ilosć elementów

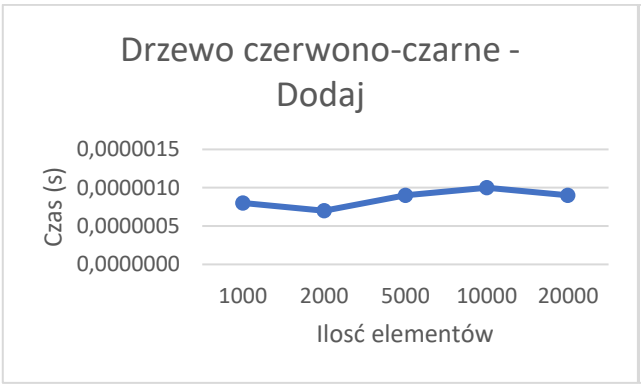
4.3 Kopiec

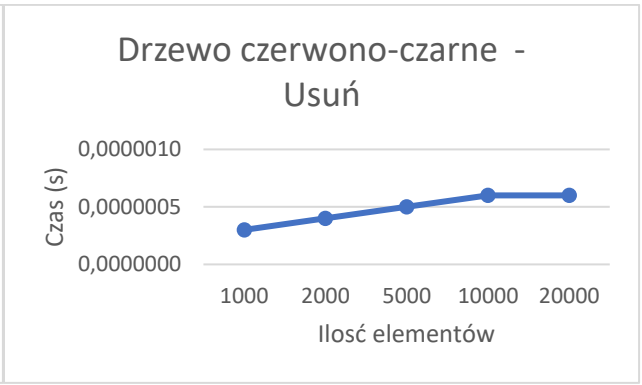


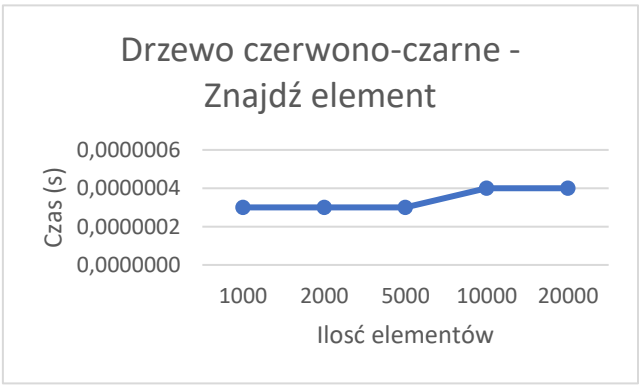




4.4 Drzewo czerwono-czarne







5 Wnioski wynikające z pomiarów

W dużej części operacje przyjmowały czas wykonania, który został wyznaczony przez stopień skomplikowania. Jednak niektóre operacje, które miały przyjmować złożoność rzędu O(1) przyjmują czas O(n) (funkcje dodawania i usuwania na kopcu), co jest zapewne spowodowane relokacją pamięci zależnej od ilości elementów w strukturze. Warto też zauważyć, że wartość elementu nie ma znaczenia w związku z operacjami na tablicy i liście, ale ma duże znaczenie w działaniach na kopcu i drzewie binarnym, co jest związane z naturą tych struktur – wartość elementu zależy od miejsca w którym zostanie dodane oraz od budowy tych drzew.

Badając czasy działania tych struktur jesteśmy w stanie powiedzieć jak je najlepiej wykorzystać. Tablica posiada jedynie adres i wartości, które mogą być z łatwością wczytane i zmienione. Lista zawiera wskaźniki na poprzednika i następnika co ułatwia orientację w zawartości, lecz wydłuża nieco czas operacji, ponieważ musi zmienić wartości wskaźników. Kopiec dzięki ustawianiu elementów wg wartości jest przydatny w zbiorach liczbowych które musimy posortować. Drzewa czerwono-czarne są trudne w implementacji, ale dzięki samoorganizacji charakteryzują się niską złożonością obliczeniową operacji wstawiania, usuwania czy wyszukiwania elementów.