Sprawozdanie z pierwszego zadania projektowego ze Struktur Danych i Złożoności Obliczeniowej

Michał Maziarz

Kwiecień 2023

Spis treści

1 Wstęp **4**

2 Złożoności obliczeniowe poszczególnych operacji na strukturach danych na podstawie literatury **4**
2.1 Tablica dynamiczna 4
2.2 Lista dwukierunkowa 4
2.3 Kopiec binarny 4
2.4 Drzewo przeszukiwań binarnych 5
2.5 Drzewo czerwono-czarne 5

3 Metodologia przeprowadzania eksperymentów **5**

4 Pomiary i wykresy tablicy dynamicznej **8**
4.1 Wstawianie na początek 8
4.1.1 Tabela 8
4.1.2 Wykres 8
4.2 Wstawianie na środek 9
4.2.1 Tabela 9
4.2.2 Wykres 9
4.3 Wstawianie na koniec 10
4.3.1 Tabela 10
4.3.2 Wykres 10
4.4 Usuwanie z początku 11
4.4.1 Tabela 11
4.4.2 Wykres 11
4.5 Usuwanie ze środka 12
4.5.1 Tabela 12
4.5.2 Wykres 12
4.6 Usuwanie z końca 13
4.6.1 Tabela 13
4.6.2 Wykres 13

5 Pomiary i wykresy listy podwójnie wiązanej **14**
5.1 Wstawianie na początek 14
5.1.1 Tabela 14
5.1.2 Wykres 14
5.2 Wstawianie na środek 15
5.2.1 Tabela 15
5.2.2 Wykres 15
5.3 Wstawianie na koniec 16
5.3.1 Tabela 16
5.3.2 Wykres 16
5.4 Usuwanie z początku 17
5.4.1 Tabela 17

5.4.2 Wykres . . . 17
5.5 Usuwanie ze środka . . . 18
5.5.1 Tabela . . . 18
5.5.2 Wykres . . . 18
5.6 Usuwanie z końca . . . 19
5.6.1 Tabela . . . 19
5.6.2 Wykres . . . 19

6 Pomiary i wykresy kopca binarnego 20
6.1 Wstawianie . . . 20
6.1.1 Tabela . . . 20
6.1.2 Wykres . . . 20
6.2 Usuwanie . . . 21
6.2.1 Tabela . . . 21
6.2.2 Wykres . . . 21

7 Pomiary i wykresy drzewa binarnych przeszukiwań 22
7.1 Wstawianie . . . 22
7.1.1 Tabela . . . 22
7.1.2 Wykres . . . 22
7.2 Usuwanie . . . 23
7.2.1 Tabela . . . 23
7.2.2 Wykres . . . 23
7.3 Przeszukiwanie . . . 24
7.3.1 Tabela . . . 24
7.3.2 Wykres . . . 24

8 Pomiary i wykresy drzewa czerwono-czarnego 25
8.1 Wstawianie . . . 25
8.1.1 Tabela . . . 25
8.1.2 Wykres . . . 25
8.2 Usuwanie . . . 26
8.2.1 Tabela . . . 26
8.2.2 Wykres . . . 26
8.3 Przeszukiwanie . . . 27
8.3.1 Tabela . . . 27
8.3.2 Wykres . . . 27

9 Wnioski 28

1 Wstęp

Pierwszym zadaniem projektowym była implementacja oraz dokonanie pomiaru czasu działania różnych kluczowych operacji następujących struktur danych:

- Tablicy dynamicznej
- Listy dwukierunkowej
- Kopca binarnego
- Drzewa binarnych przeszukiwań
- Drzewa czerwono-czarnego

Wszystkie struktury przechowywały wartości w postaci 4-bajtowej liczby całkowitej ze znakiem (int), a cały projekt został napisany w języku C++, używając paradygmatu programowania obiektowego.

Cały projekt zawiera również program, który zawiera menu umożliwiające wykonanie poszczególnych operacji na strukturach ręcznie, oraz przeprowadzenie testów

2 Złożoności obliczeniowe poszczególnych operacji na strukturach danych na podstawie literatury

2.1 Tablica dynamiczna

- Wstawianie (niezależnie gdzie): $O(n)$
- Usuwanie (niezależnie gdzie): $O(n)$

2.2 Lista dwukierunkowa

Uwaga. Moja implementacja zawiera tylko wskaźnik początku listy, nie zawiera zaś wskaźnika na koniec listy

- Wstawianie na początek: $O(1)$
- Wstawianie w inne miejsce niż początek: $O(n)$
- Usuwanie z początku: $O(1)$
- Usuwanie z innego miejsca niż początek: $O(n)$

2.3 Kopiec binarny

- Wstawianie: $O(log\ n)$
- Usuwanie maksymalnej / minimalnej wartości (zależnie od rodzaju kopca): $O(log\ n)$

2.4 Drzewo przeszukiwań binarnych

- Wstawianie: $O(h)$
- Usuwanie: $O(h)$
- Przeszukiwanie: $O(h)$

Gdzie h - wysokość drzewa

2.5 Drzewo czerwono-czarne

- Wstawianie: $O(log\ n)$
- Usuwanie: $O(log\ n)$
- Przeszukiwanie: $O(log\ n)$

3 Metodologia przeprowadzania eksperymentów

Szkielet przeprowadzania eksperymentów jest taki sam dla każdego testu. Na początek przedstawię wielkości liczbowe stałe dla testu każdej struktury:

- Zakres badanych wielkości struktur: 500, 1000, 10000, 50000, 100000, 250000, 500000
- Liczba przeprowadzonych testów dla pojedynczego rzędu wielkości, które zostaną uśrednione: 200

Zaczynając od najmniejszego zakresu, wprowadzam wartości do struktury, dopóki jej rozmiar nie będzie równy zakresowi. Później wykonujemy 200 razy zadaną operację dodawania w pętli, i mierzymy czas jej wykonania za pomocą biblioteki *std :: chrono*. Następnie w miarę możliwości usuwam dodaną losową wartość. Po wykonaniu pętli mierzę średnią arytmetyczną zmierzonych czasów, dodaję ją do tablicy, po czym przechodzę do następnego zakresu. Po zakończeniu testów dla wszystkich zakresów zapisuję wartości do pliku CSV.
Warto wspomnieć, że dla drzewa binarnych przeszukiwań i dla drzewa czerwono-czarnego zadbałem, aby klucze wstawiane do drzew były unikalne

Przykład wykonania testu na podstawie mierzenia wstawiania do drzewa czerwono-czarnego:

```
1  void DataStructuresTester::redBlackTreeInsertionTest() {
2      using namespace std::chrono;
3      std::vector<std::string> headers;
4      for (const auto &item: RANGES) {
5          headers.push_back(std::to_string(item));
6      }
7      auto *rbt = new RedBlackTree();
8      std::vector<double> testResults;
9      for (int size: RANGES) {
10         std::cout << "INSERTION IN RED BLACK TREE TEST (" << size << ")
               " << std::endl;
```

```
        while (rbt->size < size) {
            try {
                rbt->insert(sdizoUtils::getRandomInt());
            } catch (std::exception &e) {
                continue;
            }
        }
        std::cout << rbt->getRealSize() << std::endl;
        std::vector<long> measuredTimes;
        int value = sdizoUtils::getRandomInt();
        while (true) {
            try {
                rbt->insert(value);
                rbt->deleteNode(value);
                break;
            } catch (std::exception &e) {
                continue;
            }
        }
        for (int i = 0; i < TEST_REPEATS; i++) {
            auto start = steady_clock::now();
            rbt->insert(value);
            auto end = steady_clock::now();
            auto duration = duration_cast<nanoseconds>(end - start);
            long time = duration.count();
            measuredTimes.push_back(time);
            rbt->deleteNode(value);
        }
        double result = sdizoUtils::calculate_avg(&measuredTimes);
        testResults.push_back(result);
    }
    delete rbt;

    sdizoUtils::writeArrayToCsvFile(&testResults,
                                    "rbt_insert_test.csv",
                                    headers
    );
}
```

Pobieranie losowej liczby całkowitej znajduje się w funkcji zdefiniowanej w pliku *Utils.cpp* za pomocą wbudowanej biblioteki *random*

```
int getRandomInt() {
    std::random_device device;
    std::mt19937 rng(device());
    std::uniform_int_distribution<> distribution(-INT32_MAX, INT32_MAX)
        ;
```

```
5       return distribution(rng);
6   }
```

4 Pomiary i wykresy tablicy dynamicznej

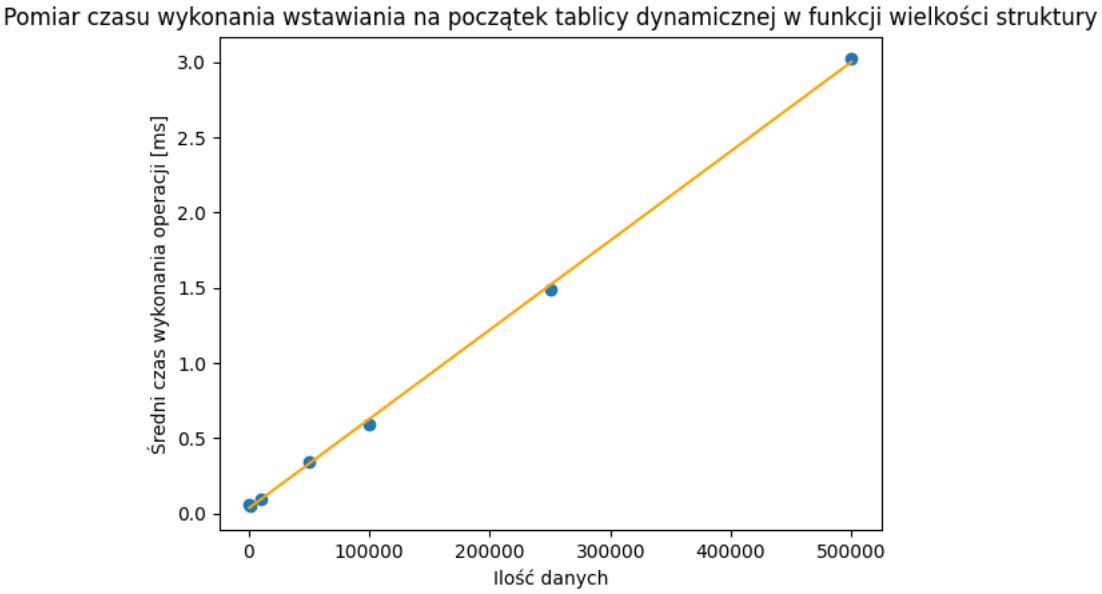
4.1 Wstawianie na początek

4.1.1 Tabela

Wstawianie na początek tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.056
1000	0.053
10000	0.092
50000	0.343
100000	0.593
250000	1.491
500000	3.021

4.1.2 Wykres



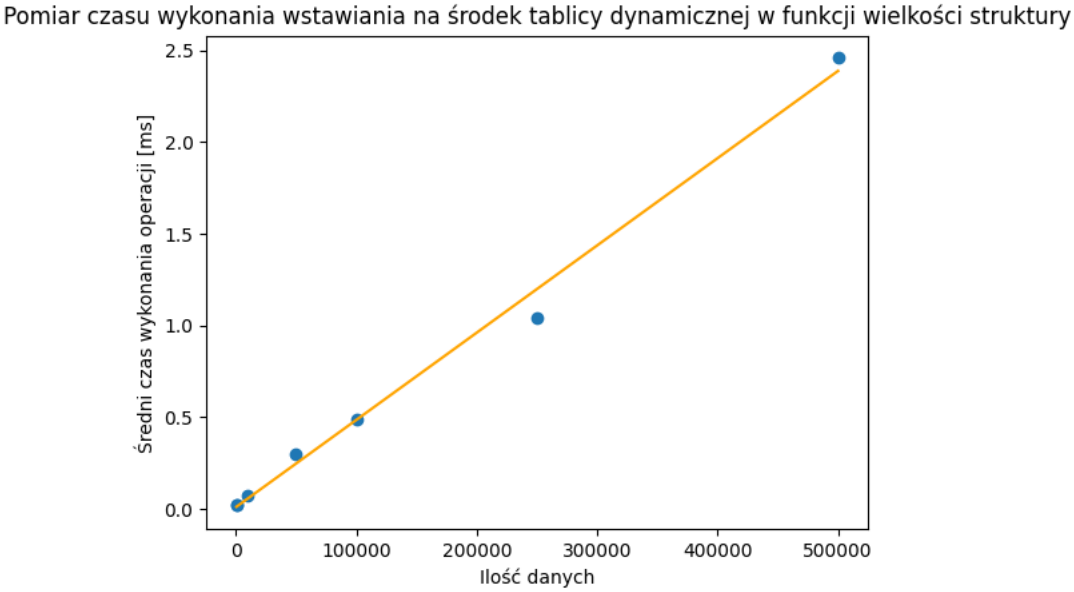
4.2 Wstawianie na środek

4.2.1 Tabela

Wstawianie na środek tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.02
1000	0.022
10000	0.071
50000	0.301
100000	0.49
250000	1.046
500000	2.458

4.2.2 Wykres



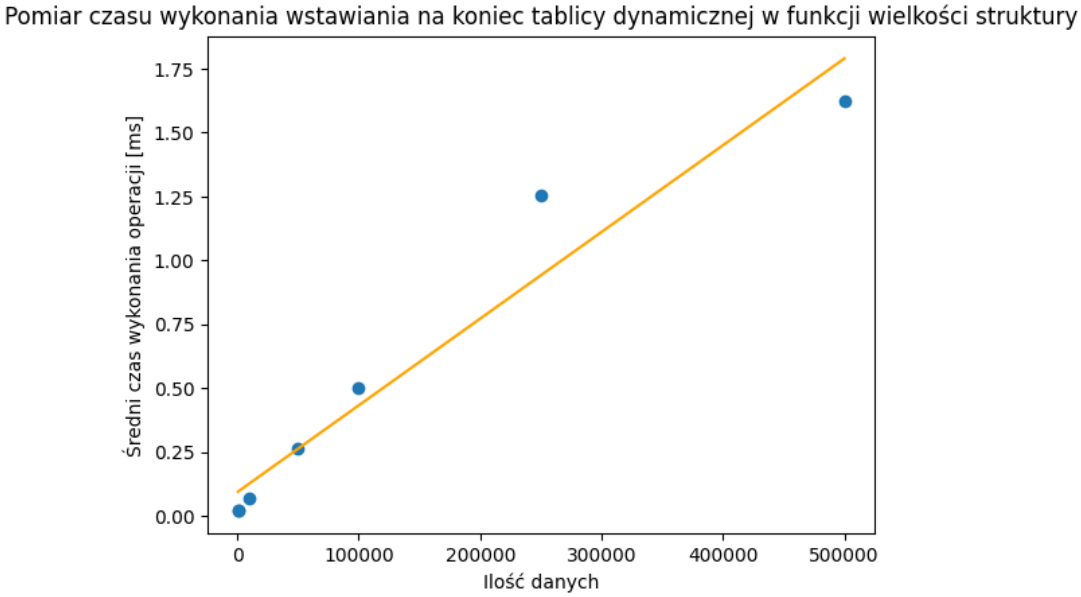
4.3 Wstawianie na koniec

4.3.1 Tabela

Wstawianie na koniec tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.019
1000	0.021
10000	0.067
50000	0.263
100000	0.499
250000	1.254
500000	1.622

4.3.2 Wykres



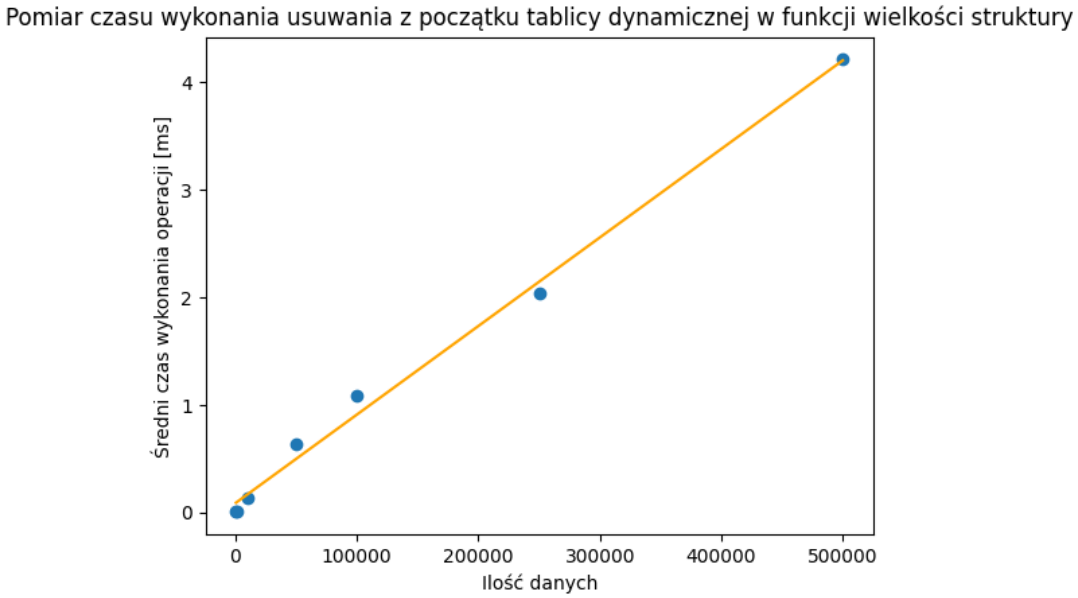
4.4 Usuwanie z początku

4.4.1 Tabela

Usuwanie z początku tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.004
1000	0.008
10000	0.128
50000	0.638
100000	1.085
250000	2.032
500000	4.209

4.4.2 Wykres



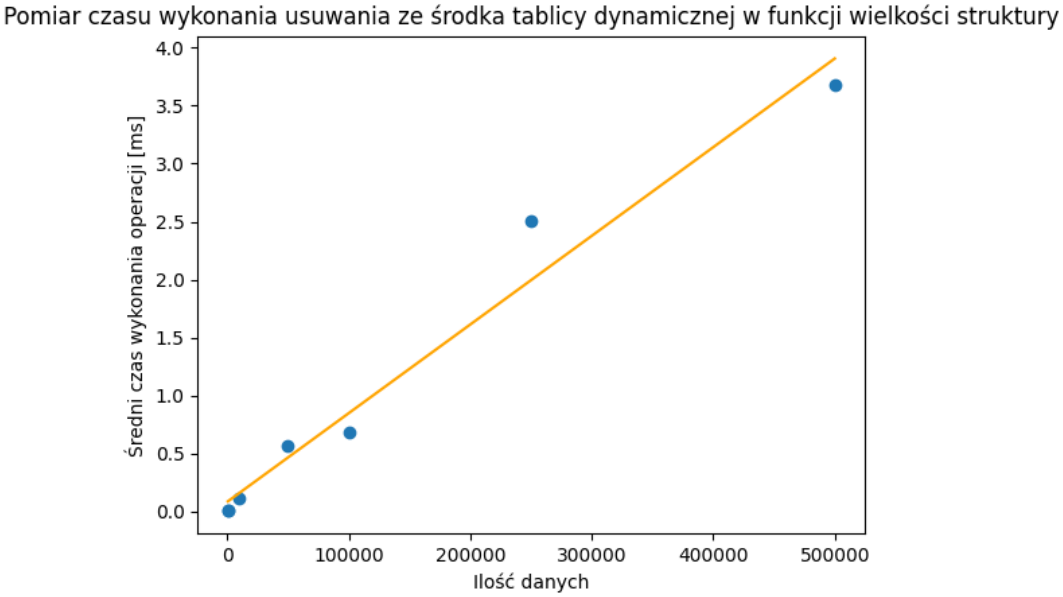
4.5 Usuwanie ze środka

4.5.1 Tabela

Usuwanie ze środka tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.003
1000	0.007
10000	0.112
50000	0.566
100000	0.681
250000	2.502
500000	3.677

4.5.2 Wykres



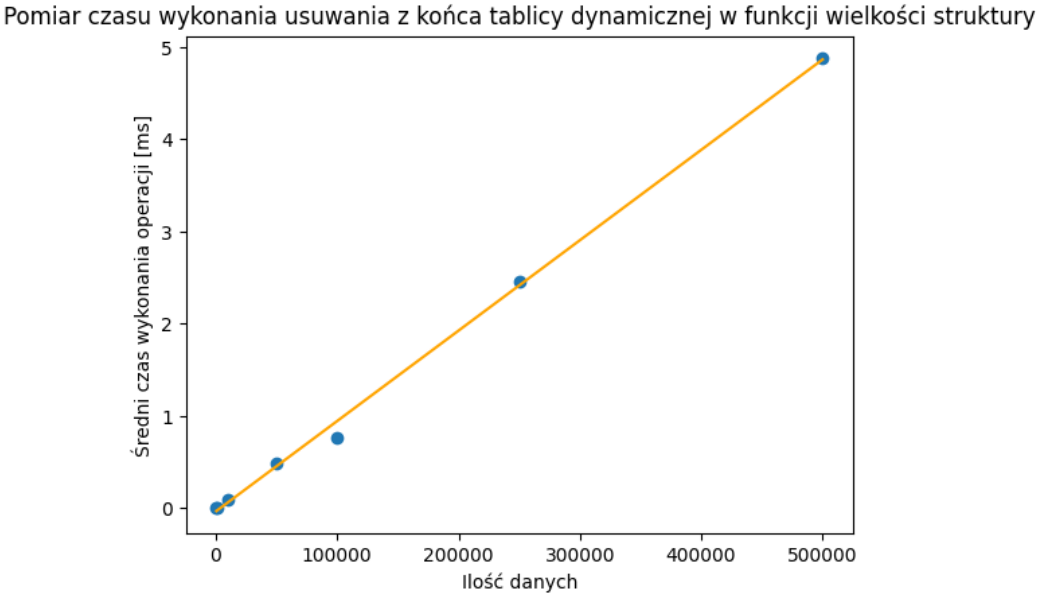
4.6 Usuwanie z końca

4.6.1 Tabela

Usuwanie z końca tablicy dynamicznej

Rozmiar struktury	czas [ms]
500	0.003
1000	0.005
10000	0.098
50000	0.483
100000	0.764
250000	2.461
500000	4.877

4.6.2 Wykres



5 Pomiary i wykresy listy podwójnie wiązanej

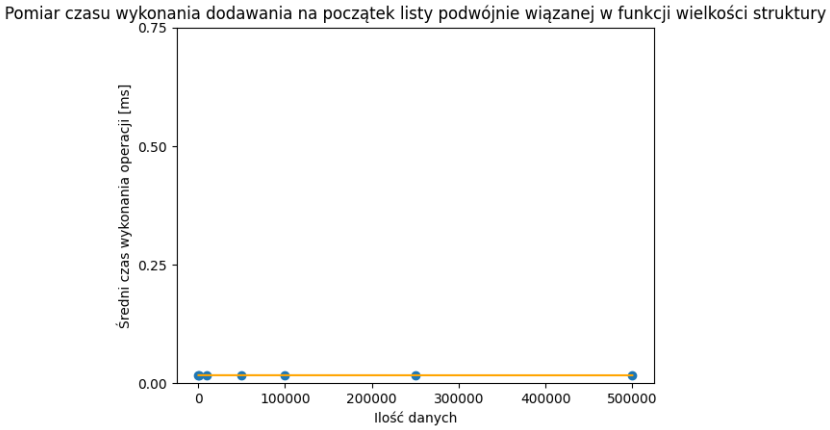
5.1 Wstawianie na początek

5.1.1 Tabela

Dodawanie na początek listy podwójnie wiązanej

Rozmiar struktury	czas [ms]
500	0.017
1000	0.017
10000	0.017
50000	0.017
100000	0.017
250000	0.017
500000	0.017

5.1.2 Wykres



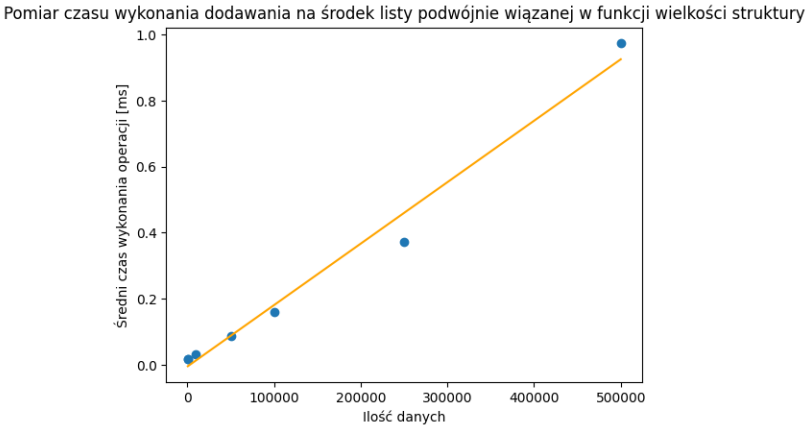
5.2 Wstawianie na środek

5.2.1 Tabela

Dodawanie na środek listy podwójnie wiązanej

Rozmiar struktury	czas [ms]
500	0.018
1000	0.018
10000	0.032
50000	0.088
100000	0.159
250000	0.372
500000	0.973

5.2.2 Wykres



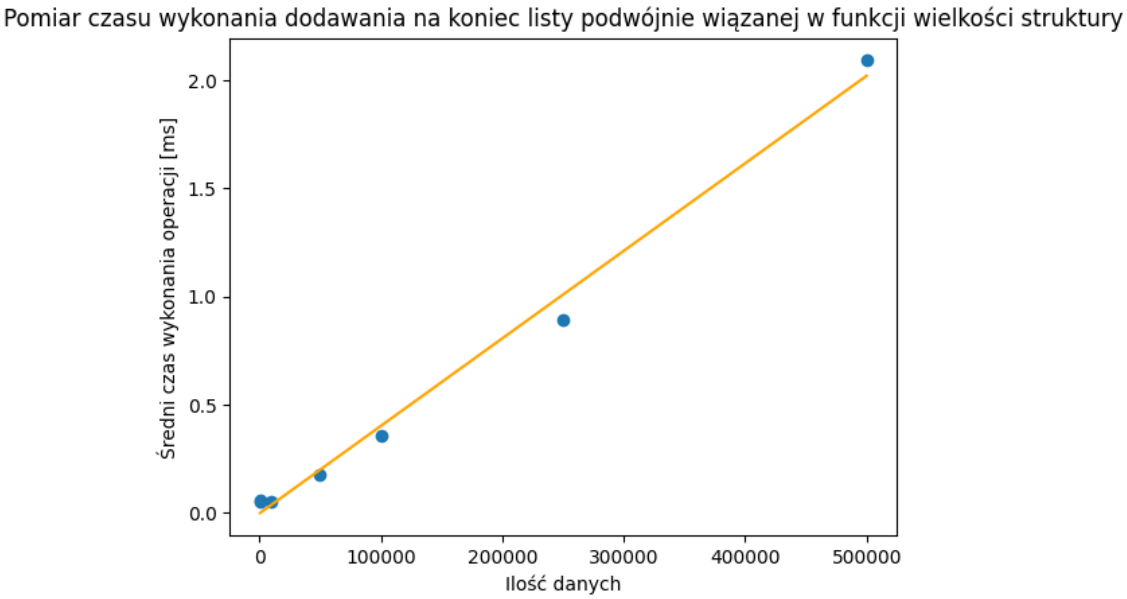
5.3 Wstawianie na koniec

5.3.1 Tabela

Dodawanie na koniec listy podwójnie wiązanej

Rozmiar struktury	czas [ms]
500	0.052
1000	0.056
10000	0.049
50000	0.173
100000	0.353
250000	0.891
500000	2.091

5.3.2 Wykres



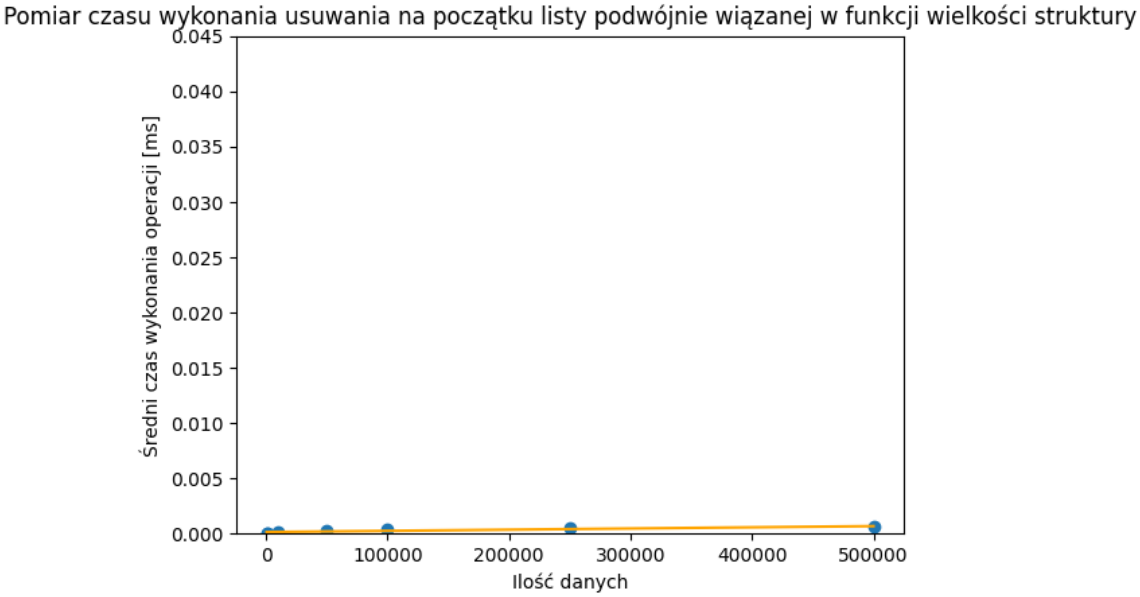
5.4 Usuwanie z początku

5.4.1 Tabela

Usuwanie z początku listy podwójnie wiązanej

Rozmiar struktury	czas [ns]
500	41.565
1000	40.185
10000	116.865
50000	268.195
100000	406.17
250000	480.14
500000	599.195

5.4.2 Wykres



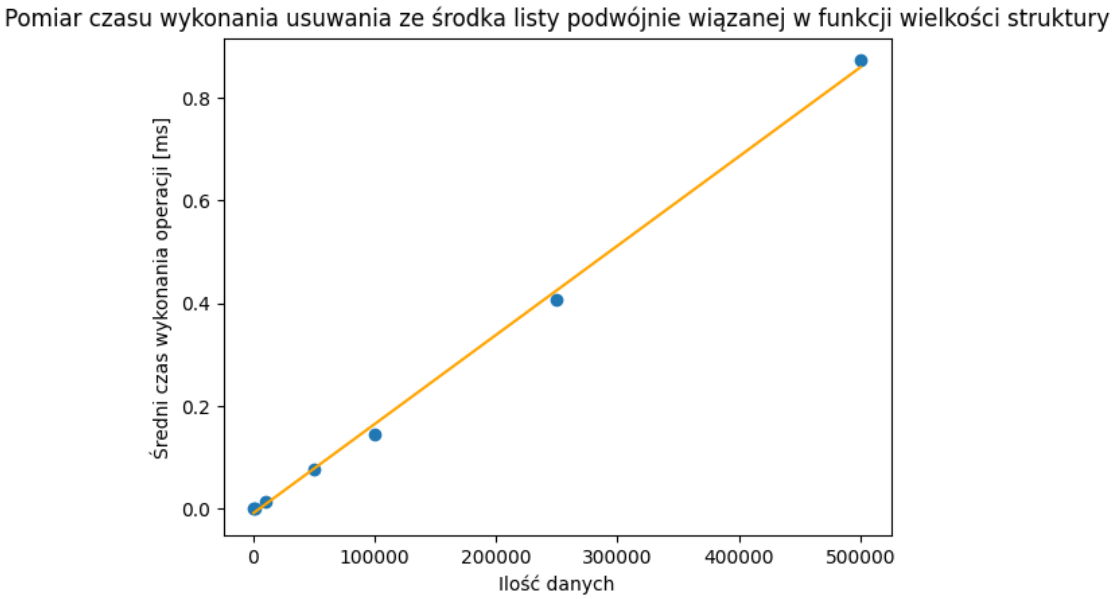
5.5 Usuwanie ze środka

5.5.1 Tabela

Usuwanie ze środka listy podwójnie wiązanej

Rozmiar struktury	czas [ms]
500	0.001
1000	0.002
10000	0.015
50000	0.077
100000	0.146
250000	0.407
500000	0.873

5.5.2 Wykres



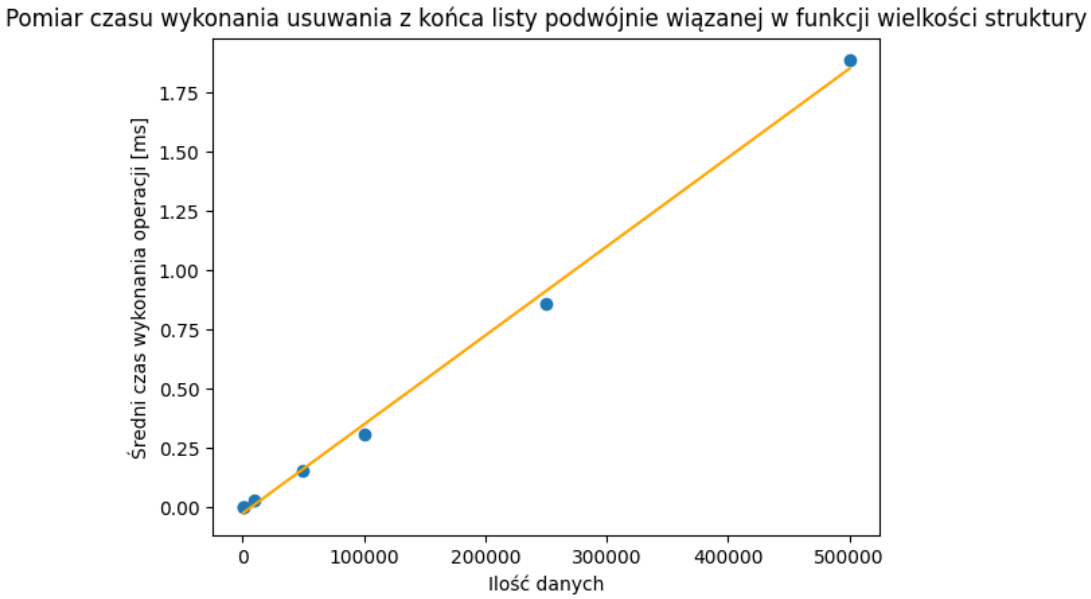
5.6 Usuwanie z końca

5.6.1 Tabela

Usuwanie z końca listy podwójnie wiązanej

Rozmiar struktury	czas [ms]
500	0.002
1000	0.003
10000	0.031
50000	0.157
100000	0.309
250000	0.863
500000	1.885

5.6.2 Wykres



6 Pomiary i wykresy kopca binarnego

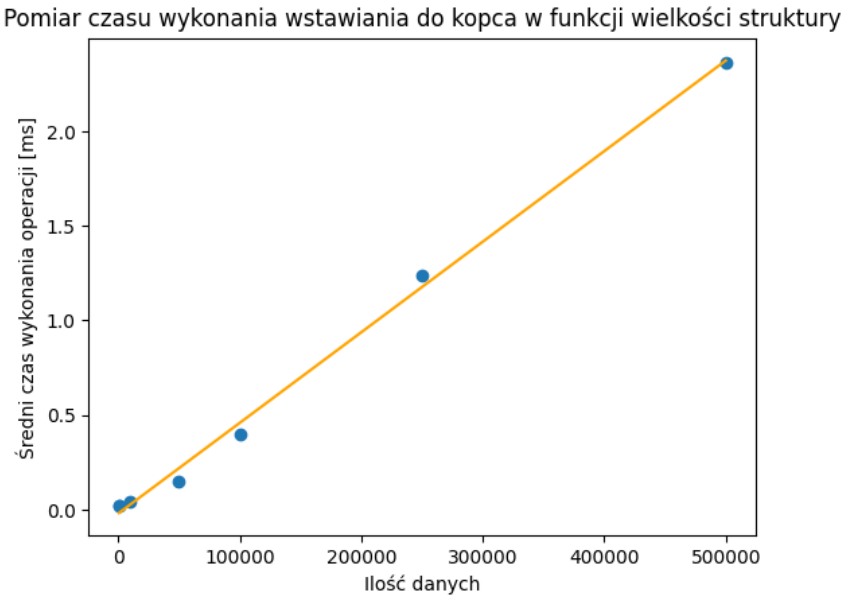
6.1 Wstawianie

6.1.1 Tabela

Wstawianie do kopca

Rozmiar struktury	czas [ms]
500	0.019
1000	0.02
10000	0.043
50000	0.146
100000	0.4
250000	1.235
500000	2.363

6.1.2 Wykres



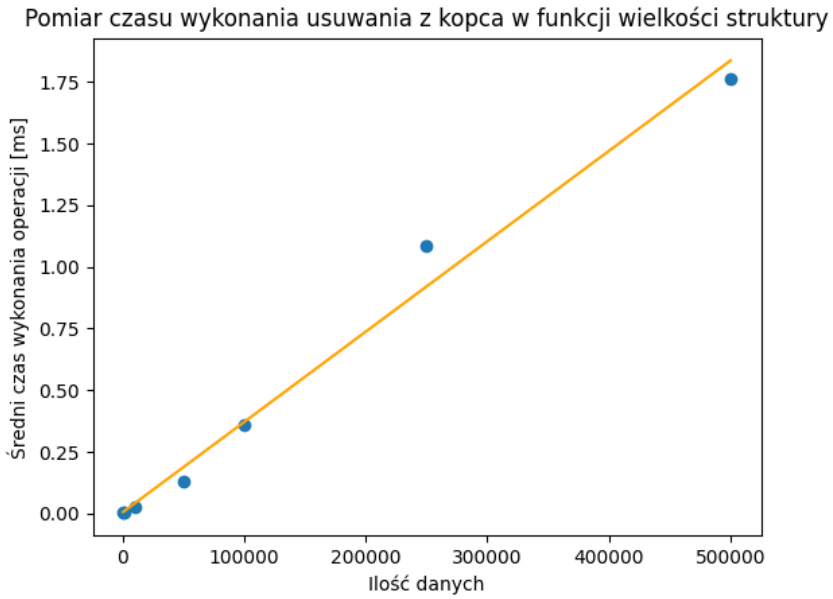
6.2 Usuwanie

6.2.1 Tabela

Usuwanie z kopca

Rozmiar struktury	czas [ms]
500	0.002
1000	0.003
10000	0.025
50000	0.132
100000	0.357
250000	1.086
500000	1.762

6.2.2 Wykres



7 Pomiary i wykresy drzewa binarnych przeszukiwań

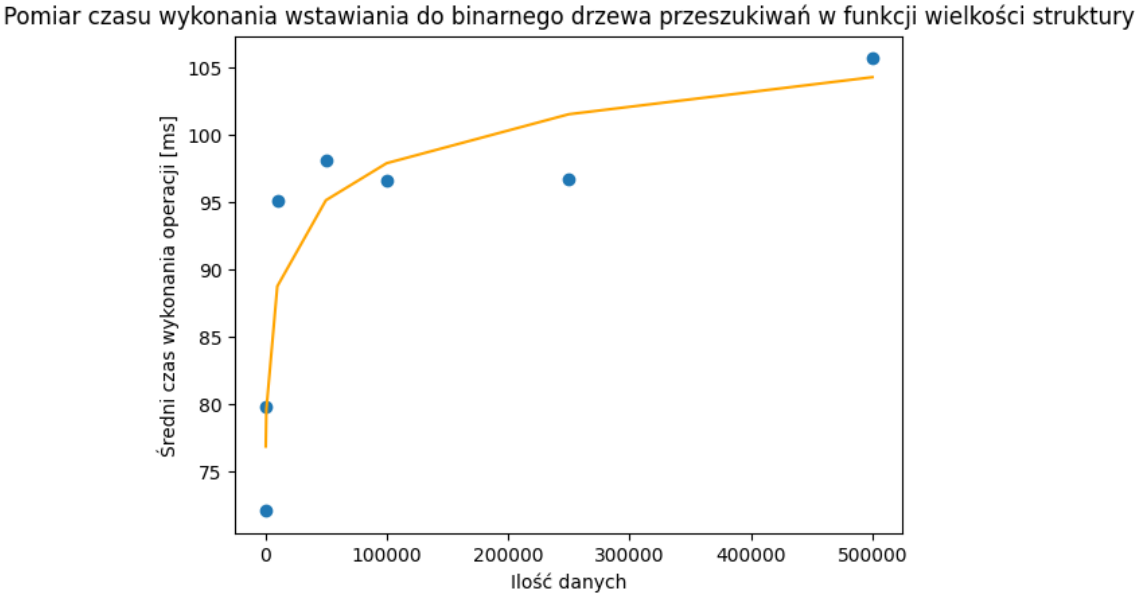
7.1 Wstawianie

7.1.1 Tabela

Wstawianie do binarnego drzewa przeszukiwań

Rozmiar struktury	czas [ns]
500	79.855
1000	72.08
10000	95.155
50000	98.125
100000	96.585
250000	96.715
500000	105.69

7.1.2 Wykres



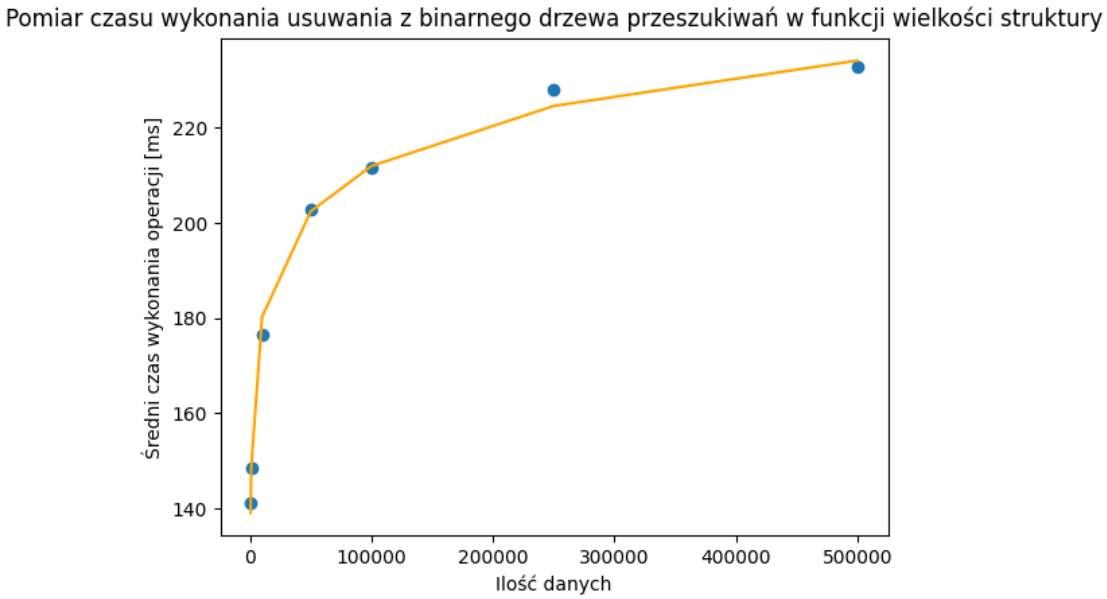
7.2 Usuwanie

7.2.1 Tabela

Usuwanie z binarnego drzewa przeszukiwań

Rozmiar struktury	czas [ns]
500	141.02
1000	148.345
10000	176.36
50000	202.745
100000	211.57
250000	227.995
500000	232.745

7.2.2 Wykres



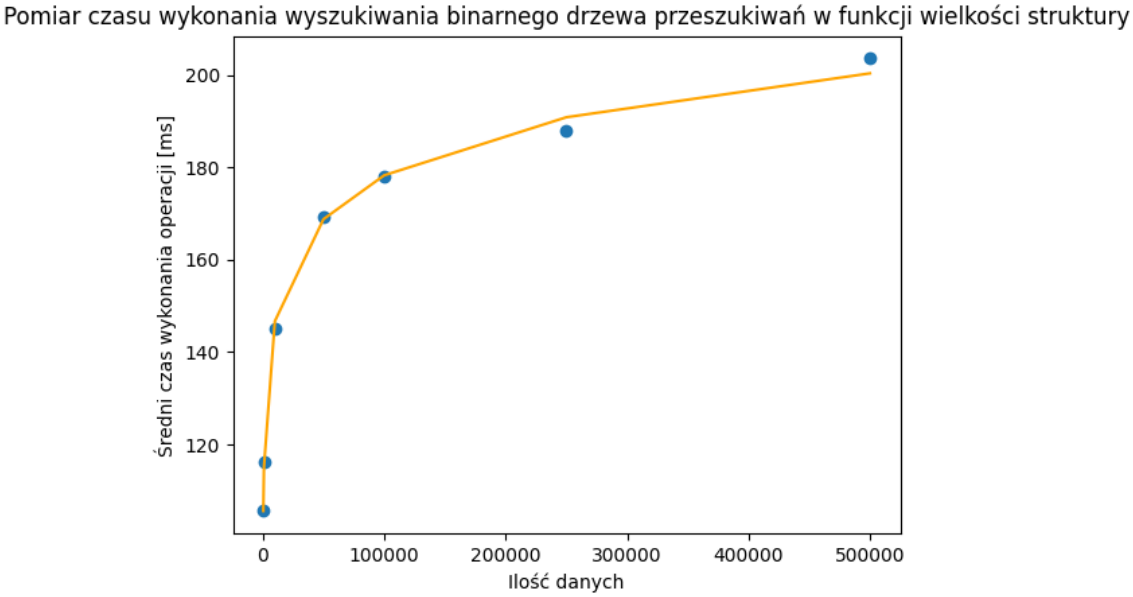
7.3 Przeszukiwanie

7.3.1 Tabela

Przeszukiwanie binarnego drzewa przeszukiwań

Rozmiar struktury	czas [ns]
500	105.67
1000	116.225
10000	144.965
50000	169.225
100000	177.895
250000	187.87
500000	203.53

7.3.2 Wykres



8 Pomiary i wykresy drzewa czerwono-czarnego

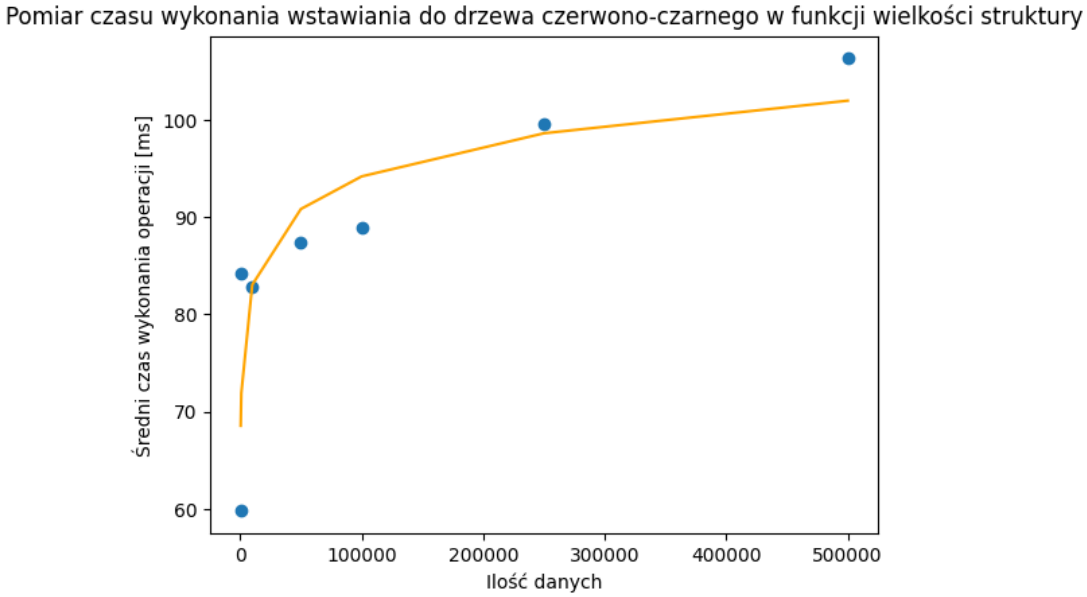
8.1 Wstawianie

8.1.1 Tabela

Wstawianie do drzewa czerwono-czarnego

Rozmiar struktury	czas [ns]
500	59.79
1000	84.26
10000	82.855
50000	87.45
100000	88.895
250000	99.49
500000	106.28

8.1.2 Wykres



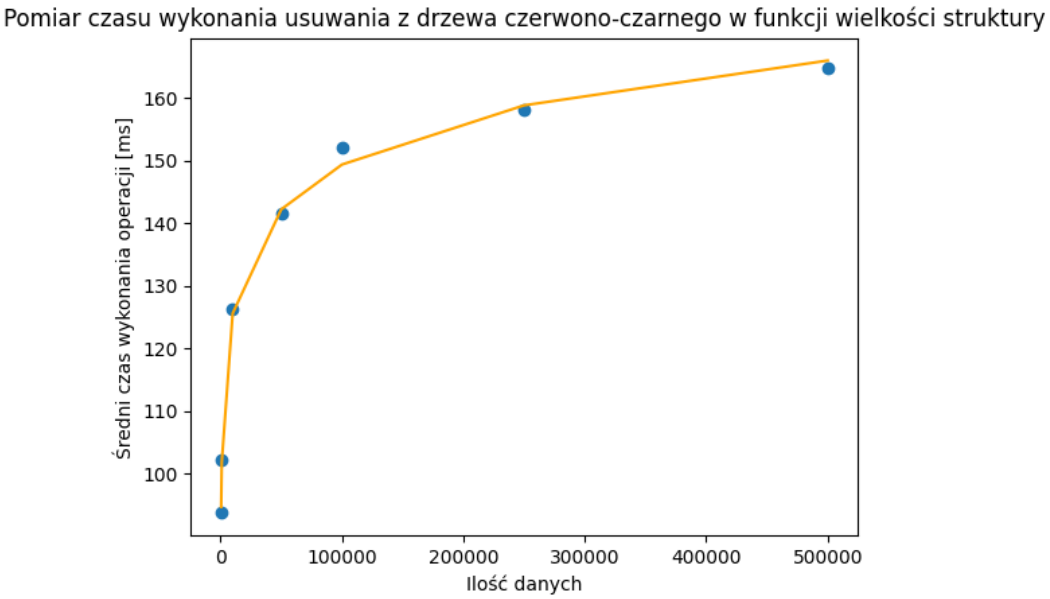
8.2 Usuwanie

8.2.1 Tabela

Usuwanie z drzewa czerwono-czarnego

Rozmiar struktury	czas [ns]
500	93.7
1000	102.075
10000	126.285
50000	141.48
100000	152.03
250000	158.045
500000	164.645

8.2.2 Wykres



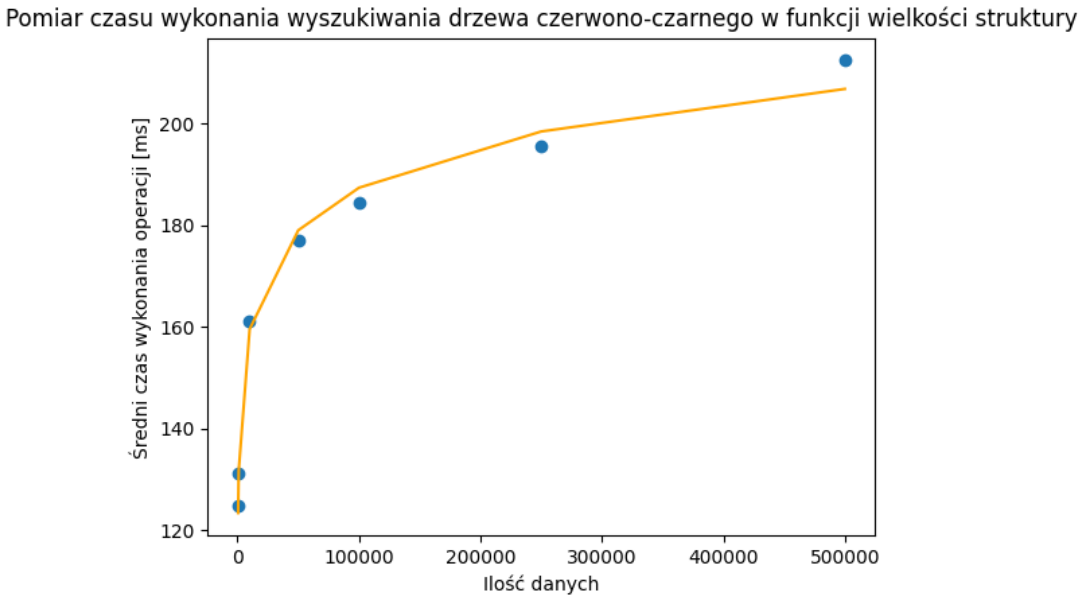
8.3 Przeszukiwanie

8.3.1 Tabela

Przeszukiwanie drzewa czerwono-czarnego

Rozmiar struktury	czas [ns]
500	124.63
1000	131.025
10000	161.185
50000	177.135
100000	184.355
250000	195.535
500000	212.455

8.3.2 Wykres



9 Wnioski

Udało się pomyślnie wykonać pomiary i wykresy dla wszystkich struktur danych.

Tablica dynamiczna jest zdecydowanie najwolniejszą strukturą. linie trendu wykresów wszystkich operacji wskazują na zakładaną złożoność liniową. Wolne czasy operacji wynikają z realokacji tablicy za każdym razem, gdy jakaś wartość była wstawiana, bądź usuwana. Dodatkowo czasy wstawiania / usuwania są dłuższe, im bliżej jesteśmy początku listy, ponieważ wstawienie na miejsce niekońcowe wymaga dodatkowego przechodzenia przez tablicę w celu zrobienia miejsca na odpowiedni indeks

Lista podwójnie wiązana podobnie wykazała przewidywane zależności. Czas wstawiania i usuwania na początek jest stały, a w inne miejsca liniowy. Podobnie jak przy tablicy dynamicznej, czas wykonania operacji, jest dłuższy, w zależności jak daleko od początku się znajdujemy. Czasy operacji jednak są ogólnie szybsze niż w tablicy

Jedyną strukturą, która nie wykazała zakładanych złożoności obliczeniowych, był kopiec binarny. Zamiast zakładanego czasu logarytmicznego, kopiec wykazał czas liniowy. Powodem takiego zachowania jest fakt, że zgodnie z zaleceniami kopiec został zaimplementowany jako tablica, która jest realokowana zawsze, gdy dodajemy lub usuwamy wartość z kopca. Same w sobie operacje mają czas logarytmiczny, lecz poza nimi wykonujemy operacje realokacji tablicy, wiec czas zgodnie z równaniem:

$$O(log\ n) + O(n) = O(max(log\ n, n)) = O(n) \tag{1}$$

staje się liniowy. Gdyby założyć stałą wielkość kopca, lub kopiec ten zaimplementować za pomocą węzłów i wskaźników na nie, czas operacji byłby logarytmiczny

Drzewa binarnych przeszukiwań i czerwono-czarne również wykazały zakładane złożoności. Należy jednak pamiętać, że drzewo czerwono-czarne jest stabilniejszą strukturą danych. Złożoność pesymistyczna drzewa binarnych przeszukiwań wynosi $O(n)$, podczas gdy drzewo czerwono-czarne zawsze ma złożoność $O(log\ n)$

Literatura

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, The MIT Press; 3rd edition (July 31, 2009)

[2] www.geeksforgeeks.org

[3] www.tutorialspoint.com