SDiZO Projekt nr 1

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

Autor
Damian Łukasiewicz

Spis treści

1 Wstęp

Celem projektu było zaimplementowanie oraz dokonanie pomiaru czasu działania operacji takich jak dodawanie elementu, usunięcie elementu i wyszukanie elementu w wybranych strukturach danych.

Złożoność obliczeniowa, czyli ilość zasobów potrzebnych do wykonania pewnego algorytmu możemy podzielić na pamięciową oraz czasową. Złożoność obliczeniowa czasowa jest ilością czasu niezbędnego do rozwiązania problemu w zależności od liczby danych wejściowych. Możemy ją zatem zapisać jako funkcję liczby danych wejściowych:

$$T(n) = f(n)$$

W zadaniu zmierzono złożoność czasową w następujących strukturach:

- tablica
- lista dwukierunkowa
- kopiec binarny (element maksymalny w korzeniu)
- drzewo przeszukiwań binarnych (BST)

Wszystkie przedstawione poniżej złożności czasowe rozpatrzone są pod względem najgorszych przypadków.

1.1 Tablica

Tablica została zaimplementowana w taki sposób, że przy każdym dodawaniu oraz usuwaniu elementów jest relokowana dynamicznie. Zwiększa to złożność obliczeniową w porównaniu do tablicy o stałej liczbie elementów.

Operacja	Złożoność obliczeniowa czasowa
dodawanie na początek	$O(n)$
dodawanie w środek	$O(n)$
dodawanie na koniec	$O(n)$
usuwanie z początku	$O(n)$
usuwanie ze środka	$O(n)$
usuwanie z końca	$O(n)$
szukanie wartości	$O(n)$

1.2 Lista dwukierunkowa

Lista została zaimplementowana z tzw. ogonem, a więc wskaźnikiem na element ostatni. Dzięki temu złożonośc obliczeniowa czasowa operacji dodania oraz usunięcia na koniec struktury jest taka sama jak w przypadku dodania oraz usunięcia na początek struktury i jest ona stała w czasie.

Operacja	Złożoność obliczeniowa czasowa
dodawanie na początek	$O(1)$
dodawanie w środek	$O(n)$
dodawanie na koniec	$O(1)$
usuwanie z początku	$O(1)$
usuwanie ze środka	$O(n)$
usuwanie z końca	$O(1)$
szukanie wartości	$O(n)$

1.3 Kopiec binarny

W zaimplementowanym kopcu wielkość tablicy służącej do przechowywania wartości jest stała.

Operacja	Złożoność obliczeniowa czasowa
dodawanie wartości	$O(log(n))$
usunięcie wartości	$O(n \cdot log(n))$
usuwanie wartości maksymalnej	$O(log(n))$
szukanie wartości	$O(n)$
szukanie wartości maksymalnej	$O(1)$

1.4 Drzewo przeszukiwań binarnych (BST)

Operacja	Złożoność obliczeniowa czasowa
dodawanie wartości	$O(n)$
usunięcie wartości	$O(n)$
szukanie wartości	$O(n)$

1.5 Drzewo AVL

Operacja	Złożoność obliczeniowa czasowa
dodawanie wartości	$O(log(n))$
usunięcie wartości	$O(log(n))$
szukanie wartości	$O(log(n))$

1.6 Założenia techniczne

Wszystkie struktury zostały zaimplementowane w języku C++ 11. Podstawowym elementem struktur jest 4 bajtowa liczba całkowita (int). Do automatycznej kompilacji użyto narzędzia CMake.Program kompilowany był za pomoca GNU. Dodatkowo do sprawdzenia poprawności zaimplementowanych struktur użyta została biblioteka GoogleTest[1].

2 Pomiary czasowe

Do wykonania pomiarów czasowych wykorzystany został `std::chrono::high_resolution_clock`[2]

```
#include <windows.h>

void test(){
    high_resolution_clock::time_point t1;
    high_resolution_clock::time_point t2;

    t1 = std::chrono::high_resolution_clock::now();

    // operacja na strukturze

    t2 = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> timeElapsed = duration_cast<double>(t2 - t1);
}
```

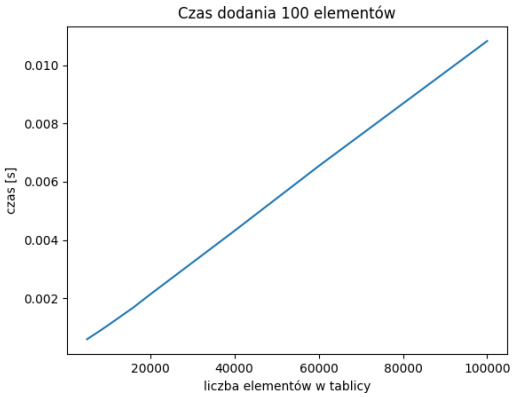
Pomiary wykonane zostały na 8 różnych wielkościach struktur. W celu uśredniania wyników dla każdego rozmiaru struktury czas został zmierzony przez daną ilość razy, a ostateczny wynik jest wartością średnią. Przykładowy test dodawania na początek tablicy:

```
    std::vector<int> sampleSizes = {5000, 8000, 10000, 16000, 20000,40000,60000,100000};
    for(auto sampleSize: sampleSizes){
        double time = 0.0;

        for(int j=0; j<100; j++) {
            // Generate random sample
            std::vector<int> sample = Random::getRandomVec(sampleSize, -1000, 1000);
            auto array = new Array();
            for (int k = 0; k < sampleSize; k++) {
                array->pushFront(sample[k]);
            }
            // Generate random values to push
            std::vector<int> randNums = Random::getRandomVec(100, -1000, 1000);

            // Start timer
            t1 = high_resolution_clock::now();
            // Push 100 new elements
            for (int k = 0; k < randNums.size(); k++) {
                array->pushFront(randNums[k]);
            }
            // Stop timer
            t2 = high_resolution_clock::now();
            duration<double> timeElapsed = duration_cast<duration<double>>(t2 - t1);
            // Add time
            time += timeElapsed.count();
        }
        double avgTime = time / 100;
        file<<sampleSize<<","<< avgTime <<std::endl;
    }
}
```

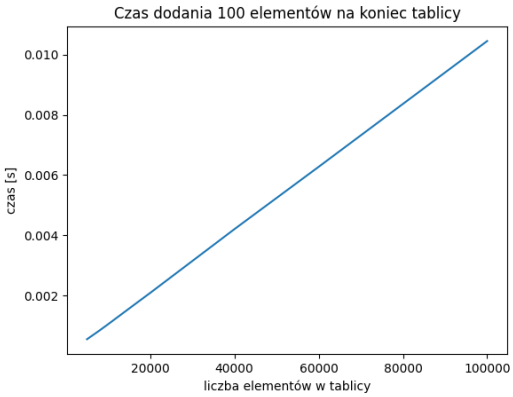
[1] https://github.com/google/googletest
[2] https://en.cppreference.com/w/cpp/chrono/high_resolution_clock

3 Wykresy

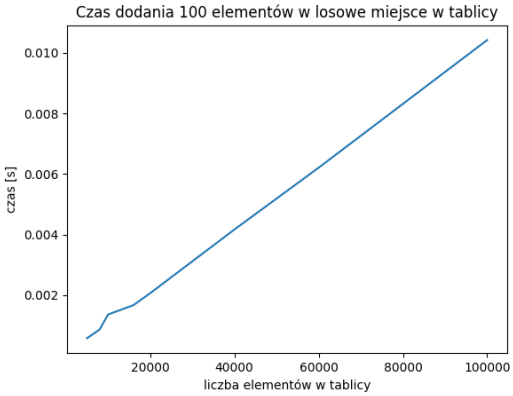
3.1 Tablica



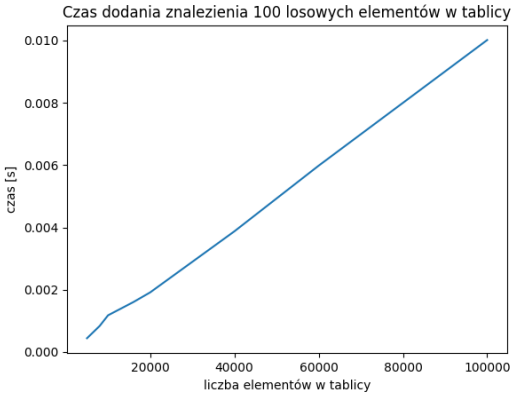
Rysunek 1: Dodawanie na początku tablicy



Rysunek 2: Dodawanie na końcu tablicy

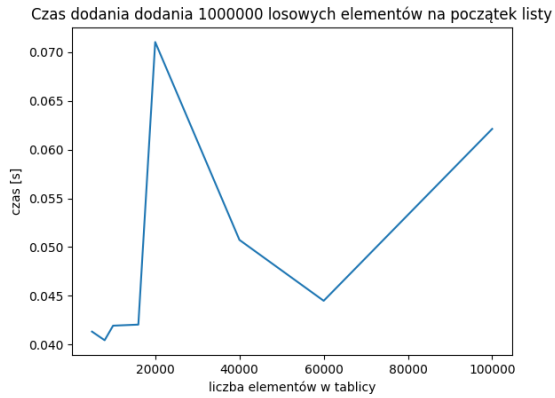


Rysunek 3: Dodanie w losowe miejsce w tablicy

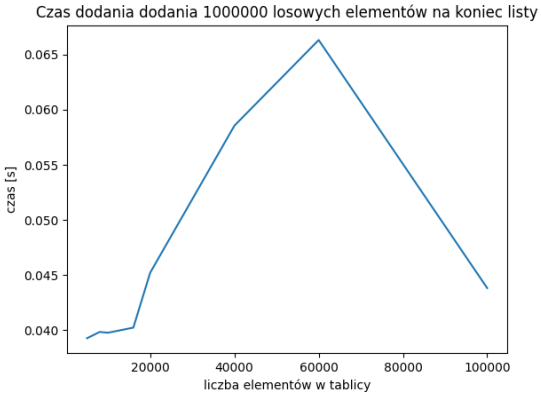


Rysunek 4: Znalezienie losowej wartości w tablicy

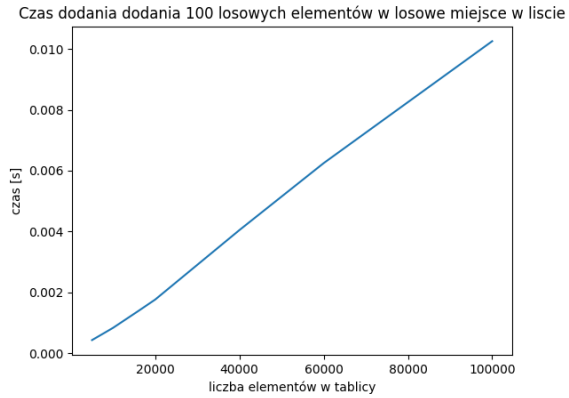
3.2 Lista dwukierunkowa



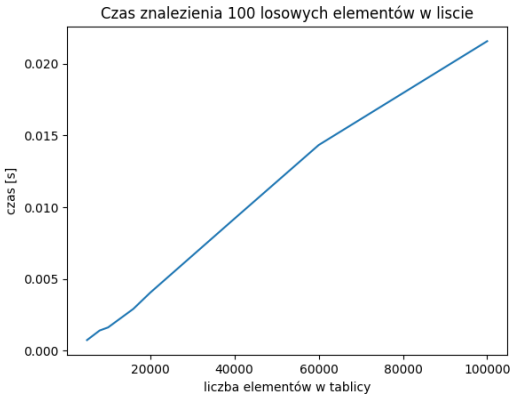
Rysunek 5: Dodawanie na początku listy



Rysunek 6: Dodawanie na końcu listy

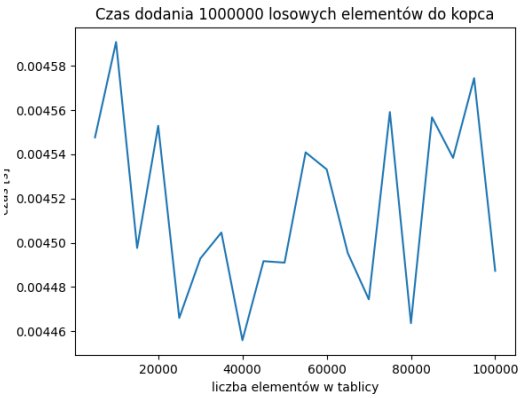


Rysunek 7: Dodanie w losowe miejsce w liście

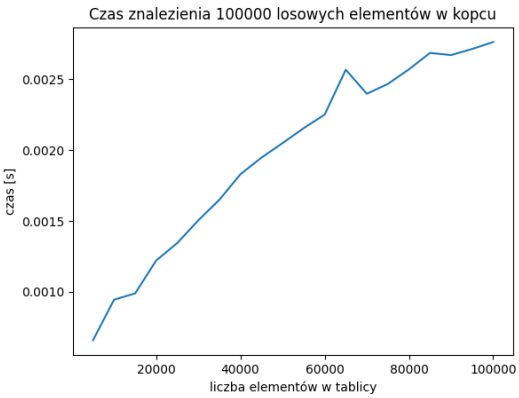


Rysunek 8: Znalezienie losowej wartości w liście

3.3 Kopiec

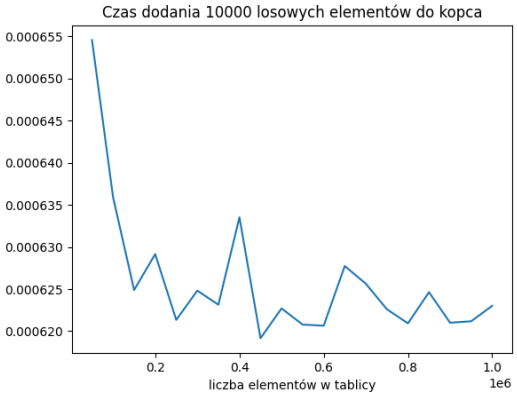


Rysunek 9: Dodawanie do kopca

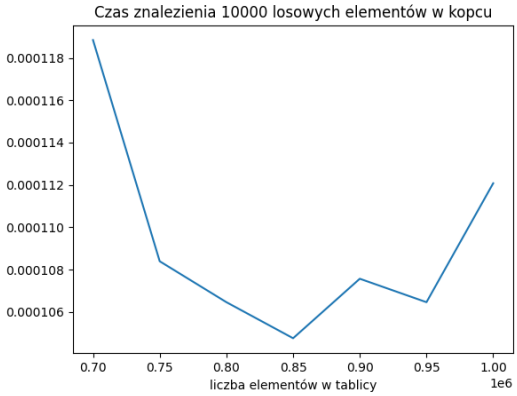


Rysunek 10: Usuwanie z kopca

3.4 BST



Rysunek 11: Dodawanie do BST



Rysunek 12: Szukanie wartości w BST

4 Wnioski

W przypadku **tablicy** alokowanej dynamicznie złożoność czasowa każdej z przedstawionych operacji zależna jest liniowo od rozmiaru tablicy.

Operacje na początku i końcu **listy** (takie jak dodawanie i odejmowanie) nie zależą od jej rozmiarów. Tak samo jak w przypadku tablicy szukanie wartości oraz dodanie elementu w losowe miejsce zależne jest od liczby elementów znajdujących się w strukturze.

Na wykresie na którym przedstawione jest dodawanie losowych wartości do **kopca** nie widać żadnej tendencji. W przypadku szukania losowych wartości widać natomiast zależność złożoności od rozmiaru struktury. Z wykresów dotyczących BST nie można jednoznacznie określić zależności złożoności czasowej od rozmiaru drzewa.