Projekt ze
Struktur Danych i Złożoności Obliczeniowiej

Jakub Grelowski - 262754
grupa K02-10d
poniedziałek P - 11:15

4 kwietnia

prowadzący: dr inż. Zbigniew Buchalski

Zadanie projektowe nr 1

Spis treści

1 Wstęp

1.1 Cel projektu

Celem projektu było napisanie programu umożliwiającego operacje na podstawowych strukturach danych oraz analizę czasową takich operacji.

1.2 Analizowane struktury danych

- Tablica dynamiczna
- Lista dwukierunkowa
- Kopiec binarny

1.3 Testowane operacje

W przypadku listy oraz tablicy:

- Dodawanie na początek
- Dodawanie na koniec
- Dodawanie na określony indeks
- Usuwanie z początku
- Usuwanie z końca
- Usuwanie z określonego indeksu
- Wyszukiwanie
- Wyświetlanie

W przypadku kopca:

- Dodawanie do kopca
- Usuwanie z korzenia
- Wyszukiwanie
- Wyświetlanie

2 Złożoność obliczeniowa

Złożoność to funkcja określająca ilość zasobów (czasowych oraz pamięciowych) wymaganych do realizacji konkretnego algorytmu. Funkcja ta zależna jest od rozmiaru wejścia algorytmu.

Oczekiwane złożności obliczeniowe testowanych operacji:

- Tablica - Dodawanie i usuwanie w każdym miejscu tablicy - $O(n)$, szukanie - $O(n)$
- Lista - Dodawanie i usuwanie na początku i końcu - $O(1)$, w innym miejscu i szukanie - $O(n)$
- Kopiec - dodawanie i usuwanie z kopca - $O(1)$, wyszkuwianie - $O(n)$

3 Implementacja

Struktury zostały zaimplementowane bez pomocy zewnętrznych bibliotek w języku C++. Przechowywanym elementem jest 4-bajtowa liczba całkowita typu *int*. Program umożliwia dwa rodzaje testów:

3.1 Testy ręczne

Testy ręczne umożliwiają samodzielne sprawdzenie poprawności działania zaimplementowanych struktur. Za pomocą konsolowego interfejsu można przetestować każdą operację oraz wyświetlić obecny stan sprawdzanej struktury. Program umożliwia ponadto wypełnienie kontenera losowymi wartości o zadanej ilości oraz zakresie.

3.2 Testy automatyczne

Testy automatyczne umożliwiają sprawdzenie, ile czasu zajmuje wykonanie każdej operacji. Po wpisaniu wielkości zbioru danych wejściowych przetestowana zostanie każda struktura, po czym na ekranie konsoli pokazany zostanie czas potrzebny na wykonanie poszczególnych działań. Pomiary czasu dokonywane są w następujący sposób:

1. Utwórz pustą strukturę
2. Dodaj x liczb z zakresu od 1 do x do struktury
3. Włącz zegar
4. Wykonaj operację (np. dodaj liczbę na początek struktury)
5. Zatrzymaj zegar
6. Zapisz wynik

Po wykonaniu stu takich pomiarów wyliczana jest z nich średnia arytmetyczna, która wyświetlana jest na ekranie konsoli. Do pomiaru czasu wykorzystano funkcję *QueryPerformanceCounter()*.

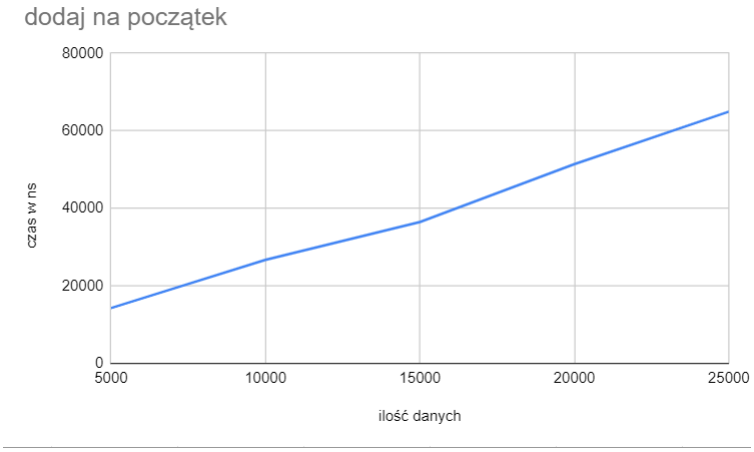
4 Wyniki pomiarów

4.1 Tablica dynamiczna

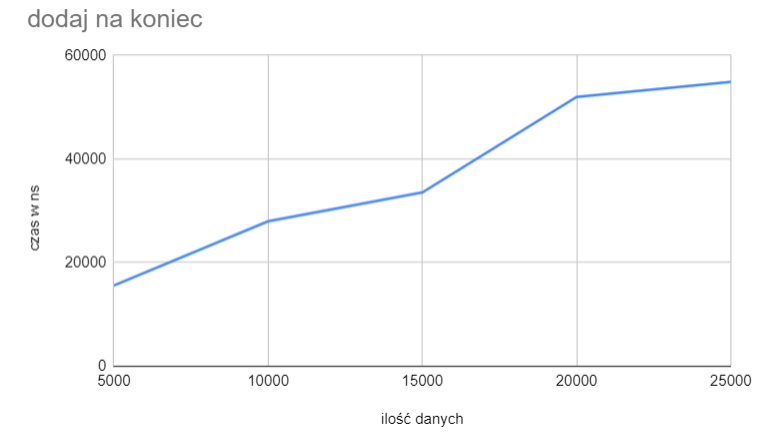
4.1.1 Tabela z wynikami

ilość danych	dodaj na początek	dodaj na koniec	dodaj do środka	usuń z początku	usuń z końca	usuń ze środka	szukaj
	ns	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*
5000	14270	15598	15172	17379	15102	14407	7162
10000	26711	27987	28173	25506	26110	27385	13715
15000	36452	33565	34706	37486	37168	40974	20898
20000	51391	52001	51981	50720	48577	51440	24805
25000	64950	54893	59477	55874	57573	62931	43472

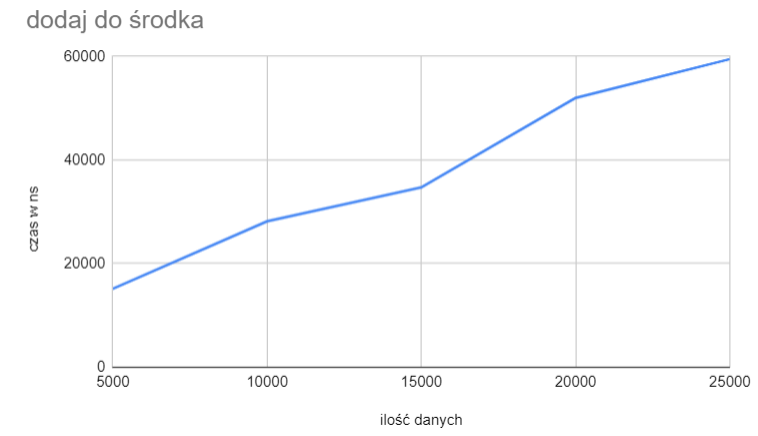
4.1.2 Wykresy



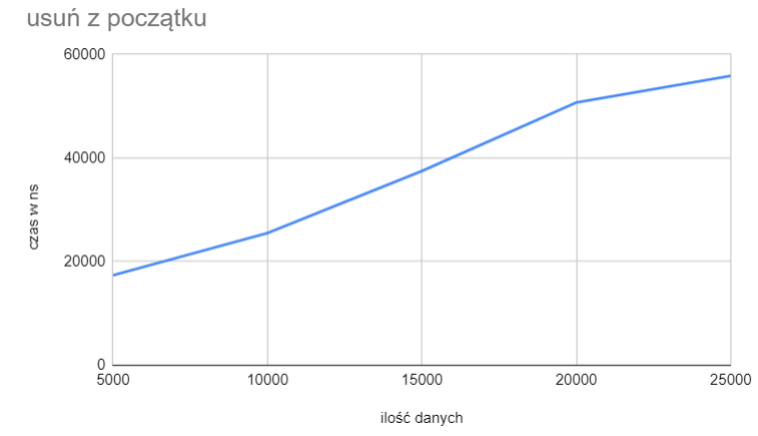
Wykres 1: Dodaj na początek - tablica



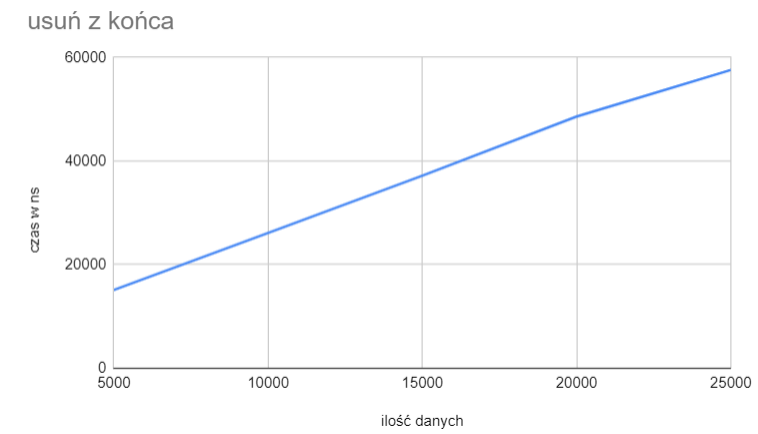
Wykres 2: Dodaj na koniec - tablica



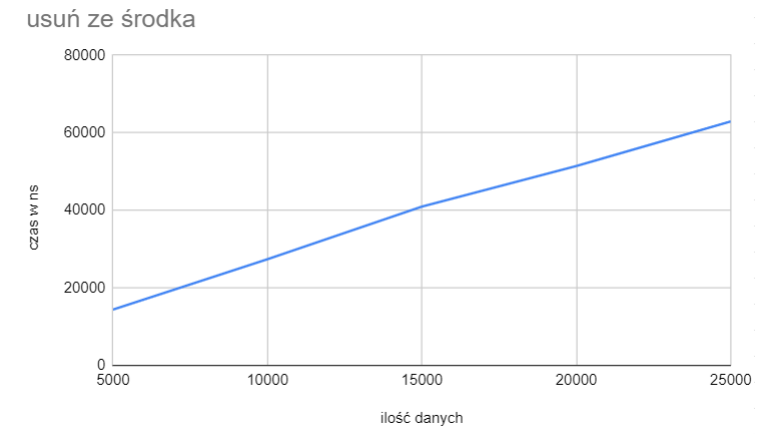
Wykres 3: Dodaj na środek - tablica



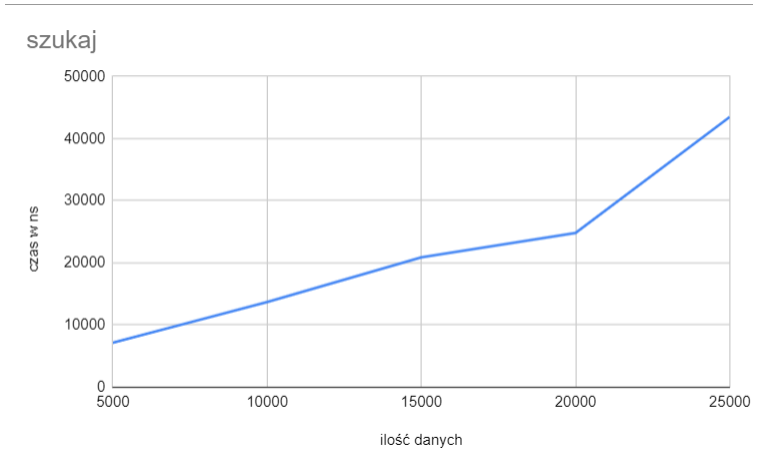
Wykres 4: Usuń z początku - tablica



Wykres 5: Usuń z końca - tablica



Wykres 6: Usuń ze środka - tablica



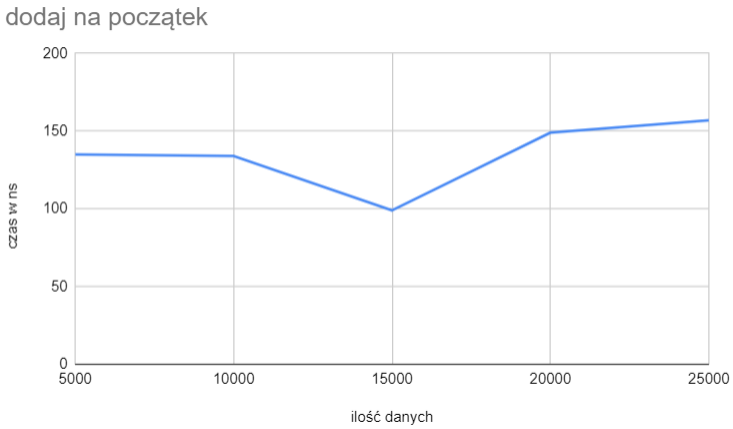
Wykres 7: Szukaj - tablica

4.2 Lista dwukierunkowa

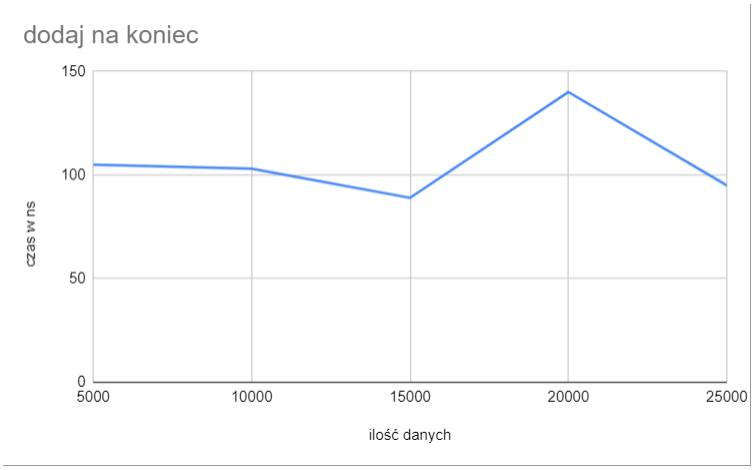
4.2.1 Tabela z wynikami

ilość danych	dodaj na początek	dodaj na koniec	dodaj do środka	usuń z początku	usuń z końca	usuń ze środka	szukaj
	ns	*ns*	*ns*	*ns*	*ns*	*ns*	*ns*
5000	135	105	28878	61	20	36751	44715
10000	134	103	99500	90	30	122699	125398
15000	99	89	143585	76	16	197513	266981
20000	149	140	215972	116	37	293516	369417
25000	157	95	282980	99	19	349771	451640

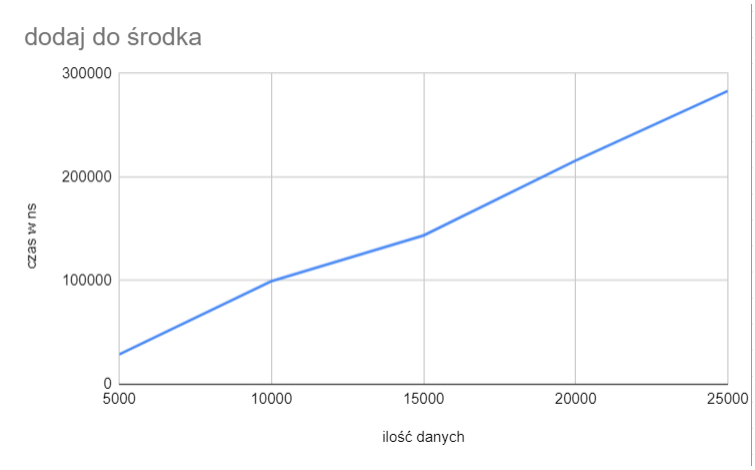
4.2.2 Wykresy



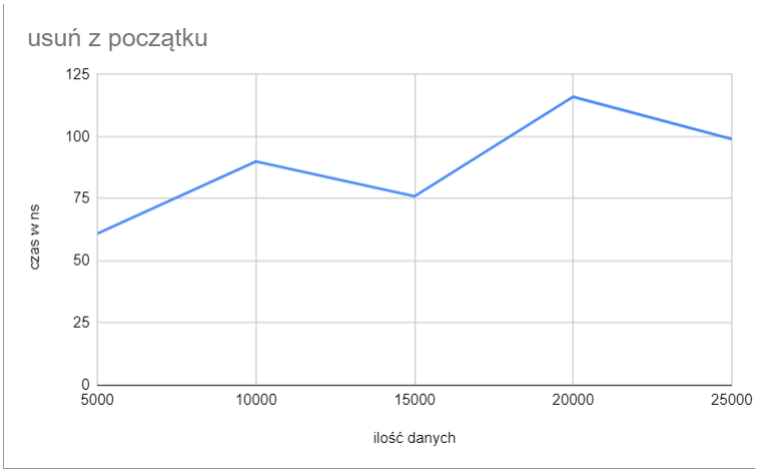
Wykres 8: Dodaj na początek - lista



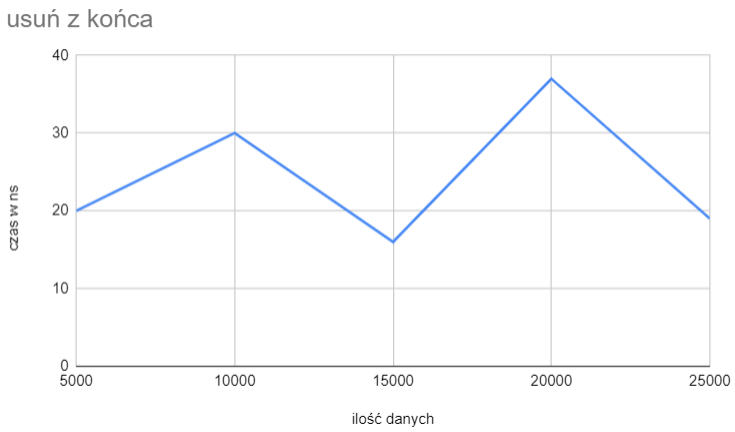
Wykres 9: Dodaj na koniec - lista



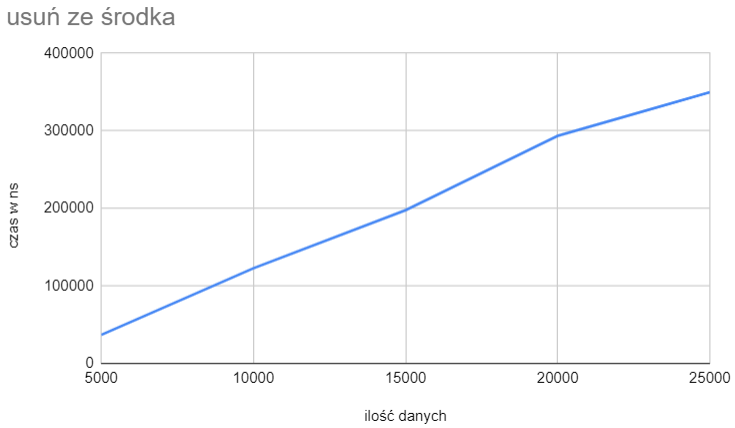
Wykres 10: Dodaj na środek - lista



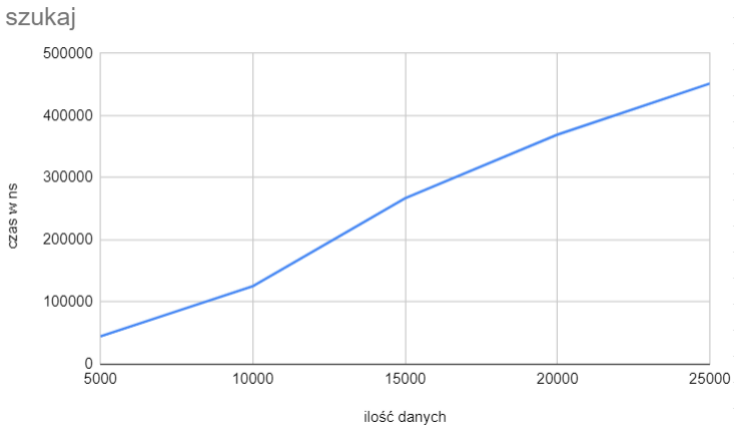
Wykres 11: Usuń z początku - lista



Wykres 12: Usuń z końca - lista



Wykres 13: Usuń ze środka - lista



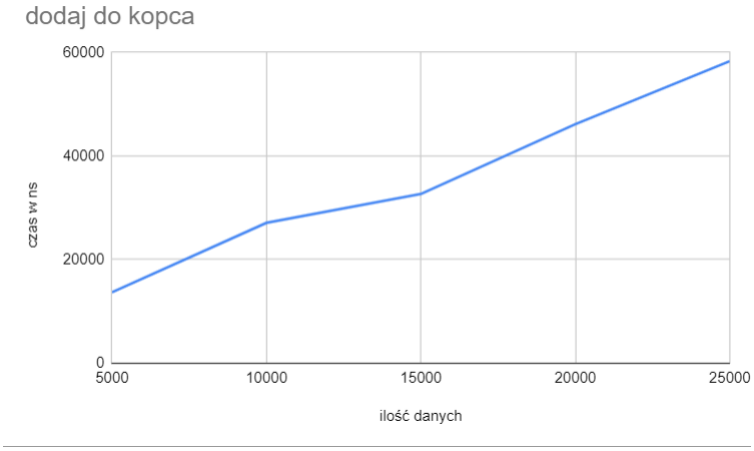
Wykres 14: Szukaj - lista

4.3 Kopiec binarny

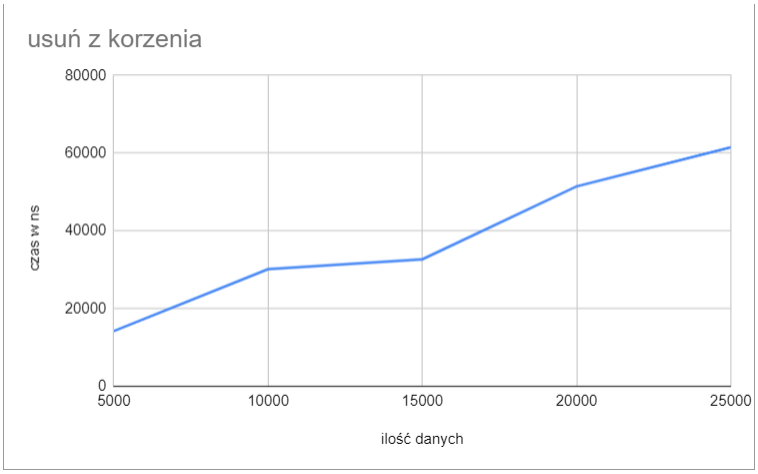
4.3.1 Tabela z wynikami

	dodaj do kopca	usuń z korzenia	szukaj
ilość danych	ns	ns	ns
5000	13712	14193	3346
10000	27097	30142	12532
15000	32667	32679	19139
20000	46132	51431	25482
25000	58293	61501	32266

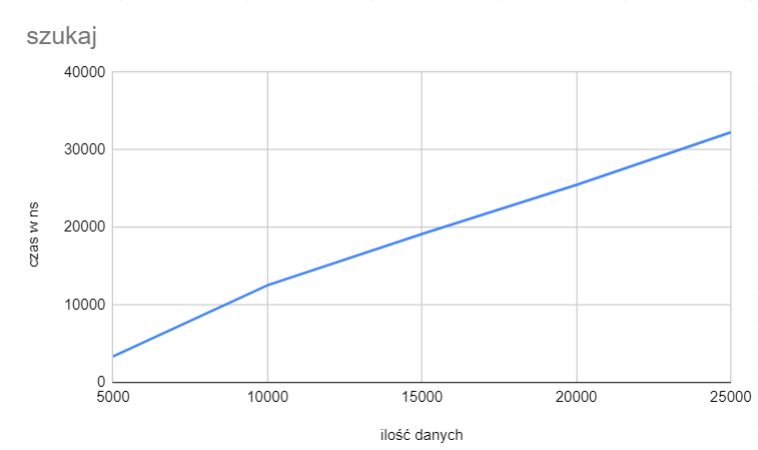
4.3.2 Wykresy



Wykres 15: Dodaj do kopca



Wykres 16: Usuń z korzenia



Wykres 17: Szukaj w kopcu

5 Wnioski

Testowane struktury zachowywały się zgodnie z oczekiwaniami. Ich złożoności czasowe w większości zgadzają się z oczekiwanymi wartościami.
Porównując tablicę oraz listę można łatwo zauważyć zarówno różnice pomiędzy nimi. Tablica będzie się lepiej nadawać do wyszukiwania wartości oraz operacjach na środku struktury, natomiast lista znacznie lepiej radzi sobie z operacjami na swoim początku i końcu (metody push i pop, a także enqueue i dequeue).
W przypadku kopca widać różnicę pomiędzy oczekiwanymi a faktycznymi wynikami. Wynikać to może z faktu iż sam kopiec jest przechowywany w dynamicznie alokowanej tablicy.