STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA

Zadanie projektowe nr 1: Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych.

5 KWIETNIA 2017

ŁUKASZ BROLL

225972

SPIS TREŚCI

WSTĘP TEORETYCZNY

Czasową złożoność obliczeniową określamy jako ilość czasu niezbędnego do rozwiązania problemu w zależności od liczby danych wejściowych. Jest to jeden z najważniejszych parametrów charakteryzujących algorytm. Decyduje on o efektywności całego programu. Podstawowymi zasobami systemowymi uwzględnianymi w analizie algorytmów są czas działania oraz obszar zajmowanej pamięci. Na złożoność czasową składają się dwie wartości: **pesymistyczna**, czyli taka, która charakteryzuje najgorszy przypadek działania oraz **oczekiwana**. Złożoność czasową oznaczamy jako O(n).

TABLICA

USUWANIE I DODAWANIE

POCZĄTEK TABLICY

Implementacja dodawania na początek tablicy polega na przypisaniu do indeksu zerowego nowej wartości i przekopiowaniu starej tablicy. Natomiast usuwanie z początku tablicy polega nam relokacji tablicy bez pierwszego elementu. Złożoność czasowa takiego algorytmu to O(n) – zależy wprost proporcjonalnie od ilości elementów. Relokacja, dzięki której korzysta się tylko z wymaganej ilości pamięci nie wpływa na złożoność czasową – czas relokacji jest wprost proporcjonalny do rozmiaru tablicy.

KONIEC TABLICY

Operacje na końcu tablicy mają stałą złożoność obliczeniową wynoszącą O(1). Z uwagi na bezpośrednią relokację tablicy podczas wykonywania funkcji, zmierzony czas wykonywania algorytmu zależy głównie od rozmiaru tablicy i czasu jej relokacji.

DOWOLNE MIEJSCE W TABLICY

Czas wykonania operacji zależy od położenia modyfikowanego elementu w tablicy. W założeniu pesymistycznym złożoność obliczeniowa wynosi O(n) tj., gdy funkcja losująca indeks zwróci wartość maksymalną - ostatni element tablicy.

WYSZUKIWANIE

W założeniu pesymistycznym złożoność obliczeniowa wynosi O(n) tj., gdy funkcja losująca indeks zwróci wartość maksymalną - ostatni element tablicy.

LISTA DWUKIERUNKOWA

USUWANIE I DODAWANIE

POCZĄTEK/KONIEC LISTY

Czas usunięcia lub dodania elementu będzie stały – złożoność obliczeniowa O(1), ponieważ wskaźniki na początek (*head) i koniec (*tail) listy gwarantują szybkie wykonanie dodawania i usuwania.

DOWOLNE MIEJSCE NA LIŚCIE

Tak, jak w przypadku listy, czas wykonania operacji zależy od położenia modyfikowanego elementu. W założeniu pesymistycznym złożoność obliczeniowa wynosi O(n) tj., gdy funkcja losująca indeks zwróci wartość maksymalną – algorytm musi wtedy dojść do poprzednika elementu i wykonać odpowiednio wstawienie lub wycięcie.

WYSZUKIWANIE

W założeniu pesymistycznym złożoność obliczeniowa wynosi O(n) tj., gdy funkcja losująca indeks zwróci wartość maksymalną - ostatni element listy.

KOPIEC

USUWANIE I DODAWANIE ELEMENTU WG KLUCZA

Głównym warunkiem wpływającym na złożoność obliczeniową operacji na kopcu jest spełnienie warunku kopca. Zarówno dodanie, jak i usunięcie elementu może wymagać odtworzenia struktury, zatem w pesymistycznym przypadku złożoność obliczeniowa zależy od liczby poziomów drzewa tj. $O(n*\log_2 n)$. Jeśli warunek kopca nie jest naruszony, wtedy złożoność czasowa jest stała – dodawany element zostaje na ostatnim liściu, co nie jest regułą w przypadku usuwania.

WYSZUKIWANIE

Jako, że struktura kopca zapisana jest w tablicy, to w założeniu pesymistycznym złożoność obliczeniowa wynosi O(n) tj., gdy funkcja losująca indeks zwróci wartość maksymalną - ostatni element tablicy.

DRZEWO RBT

Zrównoważona struktura drzewa gwarantuje, że drzewo o n wewnętrznych węzłach nigdy nie będzie miało więcej poziomów niż $2*\lg_2(n-1)$, więc wszystkie operacje wykonywane na drzewie mają pesymistyczną złożoność czasową $O(\lg_2 n)$.

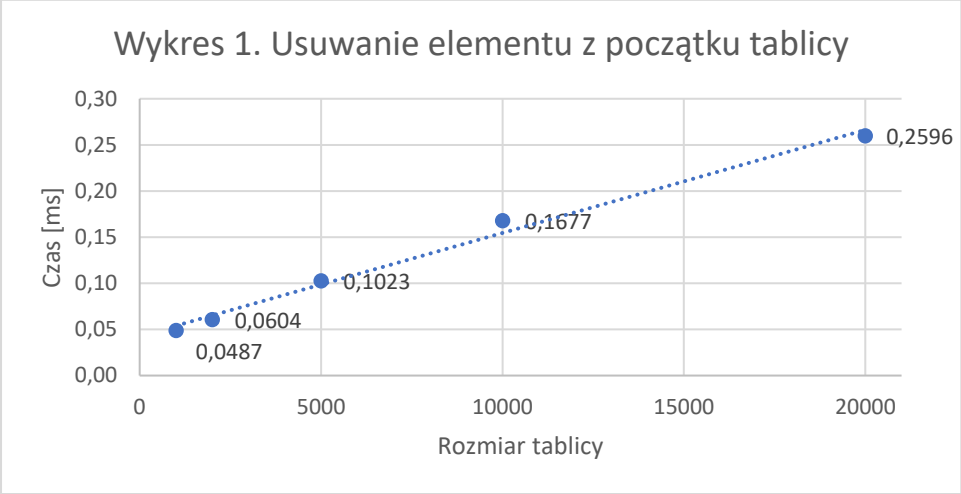
PLAN EKSPERYMENTU

- Tablica jest dynamiczna i relokowana przy usuwaniu i dodawaniu elementów, a relokacja jest uwzględniana przy każdym pomiarze czasu.
- Tablica elementów kopca jest dynamicznie tworzona z odpowiednim zapasem, ale nie jest relokowana przy każdej operacji.
- Każda operacja jest testowana na strukturze zawierającej odpowiednio 1000, 2000, 5000, 10000 i 20 000 elementów. Dla każdego pomiaru generowana jest nowa populacja. Wyniki ze 100 pomiarów są uśredniane. Wyjątek stanowi drzewo RBT, gdzie dodatkowo przeprowadzono operacje na strukturze 50000, 100000 i 200000, celem lepszej prezentacji wyników.
- Elementem struktur jest liczba typu *integer* generowana losowo z maksymalnego obsługiwanego zakresu < -16383 ; 16383 > za pomocą funkcji *rand() ,*
- Pomiarów czasu dokonano za pomocą funkcji:
  *BOOL QueryPerformanceCounter (__out LARGE_INTEGER *lpPerformanceCount);*
  Źródło licznika: http://cpp0x.pl/forum/temat/?id=21331
- Do pomiaru czasu nie były wliczane funkcje generujące dane.
- Funkcje wyszukujące losowe dane w strukturach zapisywały wyniki tylko dla tych elementów, które znajdowały się w strukturze.
- Wyświetlanie struktury drzewa zostało zrealizowane na gotowym szablonie.
  Źródło szablonu: http://eduinf.waw.pl/inf/alg/001_search/0112.php

WYNIKI EKSPERYMENTU

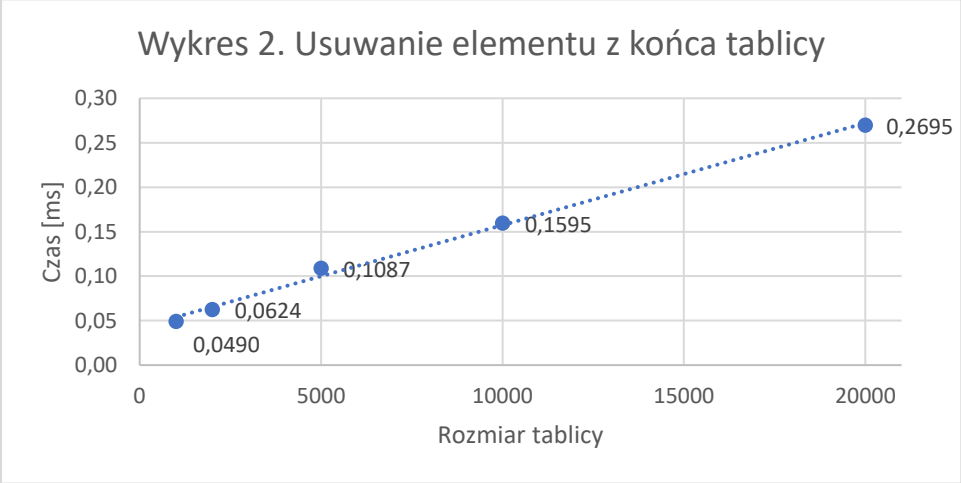
OPERACJE NA TABLICY

USUWANIE Z POCZĄTKU TABLICY



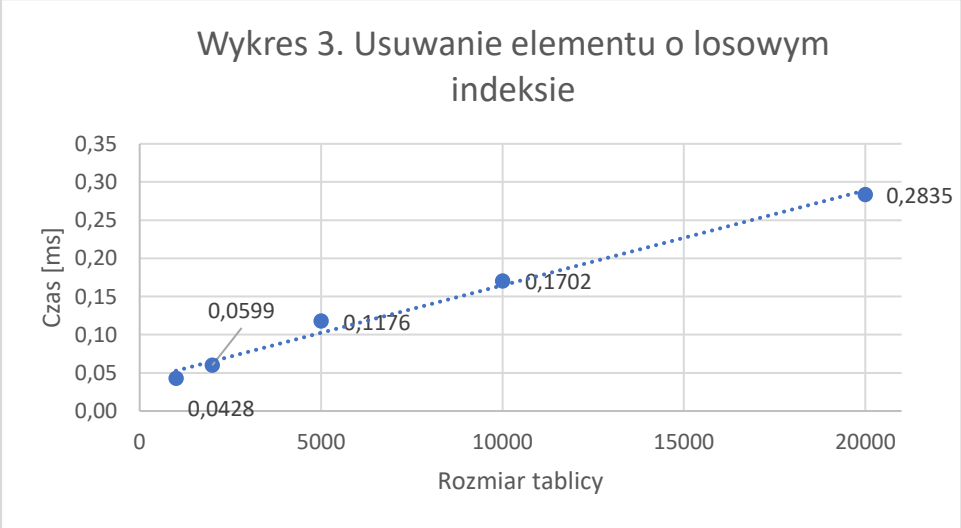
Rozmiar	czas [ms]
1000	0,0487
2000	0,0604
5000	0,1023
10000	0,1677
20000	0,2596

USUWANIE Z KOŃCA TABLICY



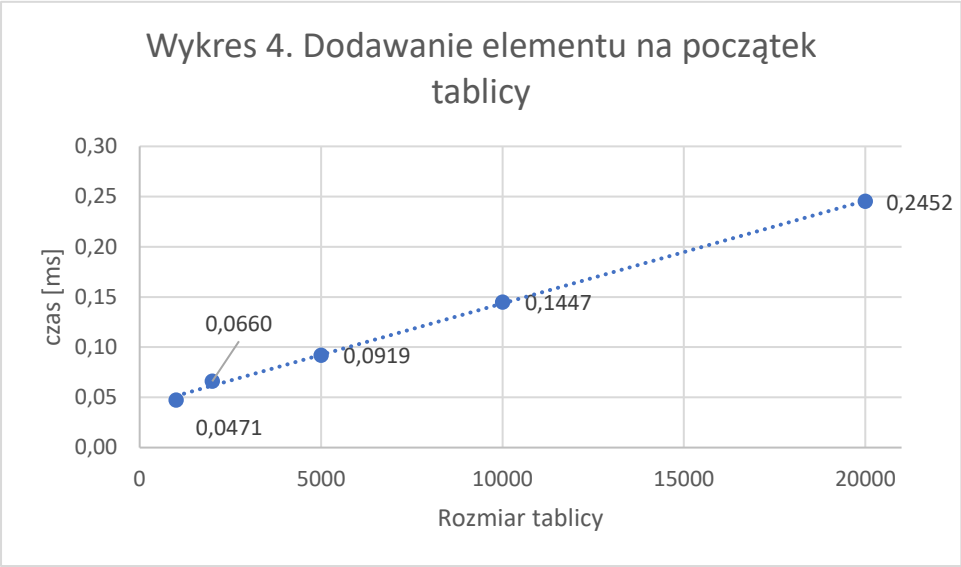
rozmiar	czas [ms]
1000	0,0490
2000	0,0624
5000	0,1087
10000	0,1595
20000	0,2695

USUWANIE LOSOWEGO ELEMENTU TABLICY



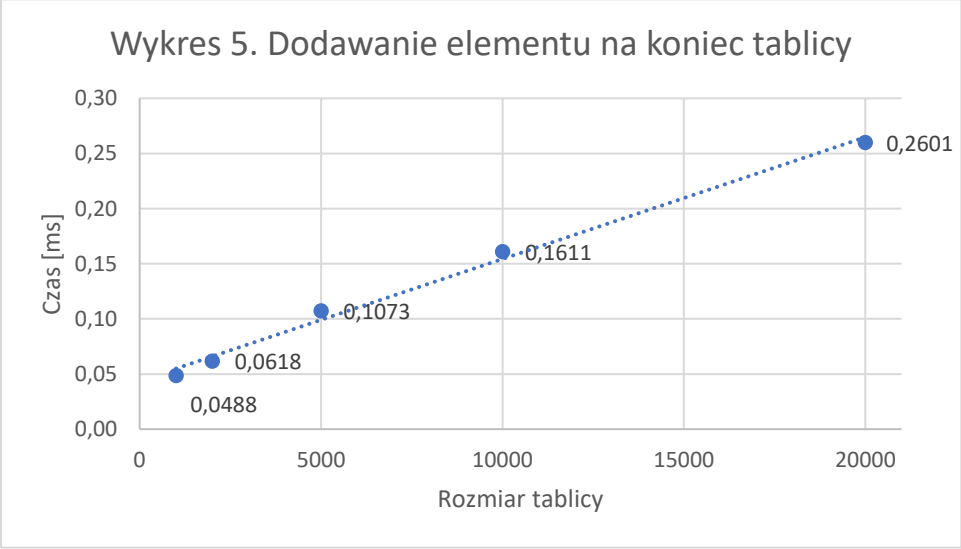
rozmiar	czas [ms]
1000	0,0428
2000	0,0599
5000	0,1176
10000	0,1702
20000	0,2835

DODAWANIE ELEMENTU NA POCZĄTEK TABLICY



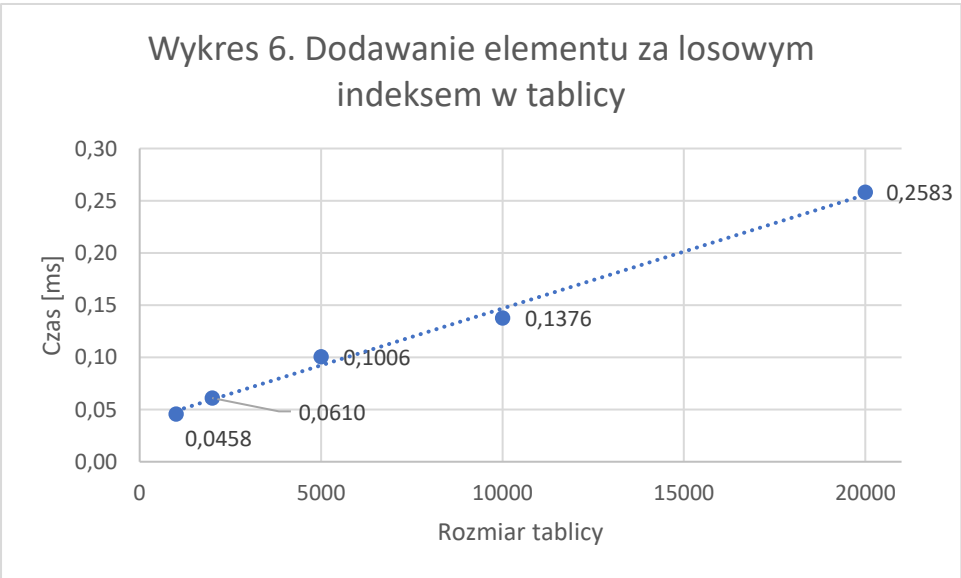
rozmiar	czas [ms]
1000	0,0471
2000	0,0660
5000	0,0919
10000	0,1447
20000	0,2452

DODAWANIE ELEMENTU NA KONIEC TABLICY



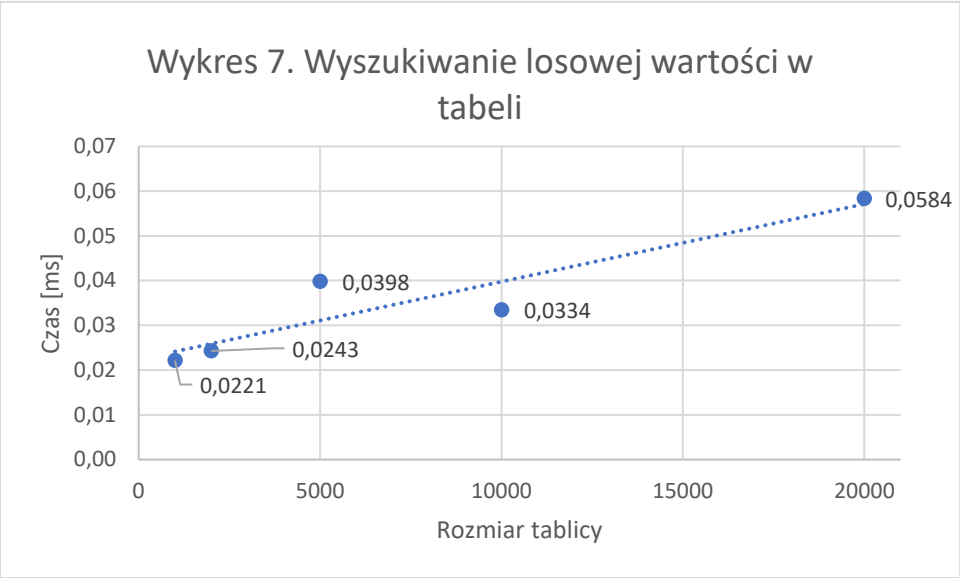
rozmiar	czas[ms]
1000	0,0488
2000	0,0618
5000	0,1073
10000	0,1611
20000	0,2601

DODAWANIE W LOSOWYM MIEJSCU ELEMENTU W TABLICY



rozmiar	czas[ms]
1000	0,0458
2000	0,0610
5000	0,1006
10000	0,1376
20000	0,2583

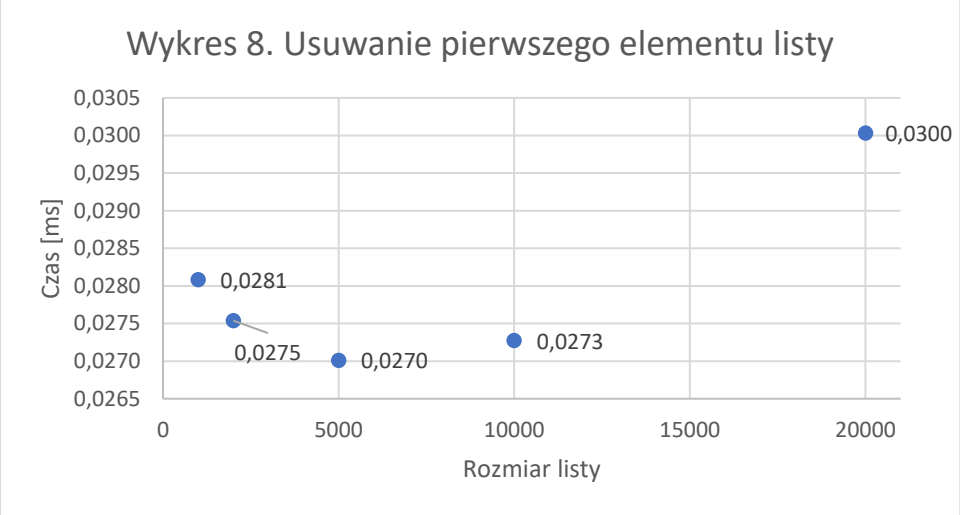
WYSZUKIWANIE W TABLICY



rozmiar	czas[ms]
1000	0,0221
2000	0,0243
5000	0,0398
10000	0,0334
20000	0,0584

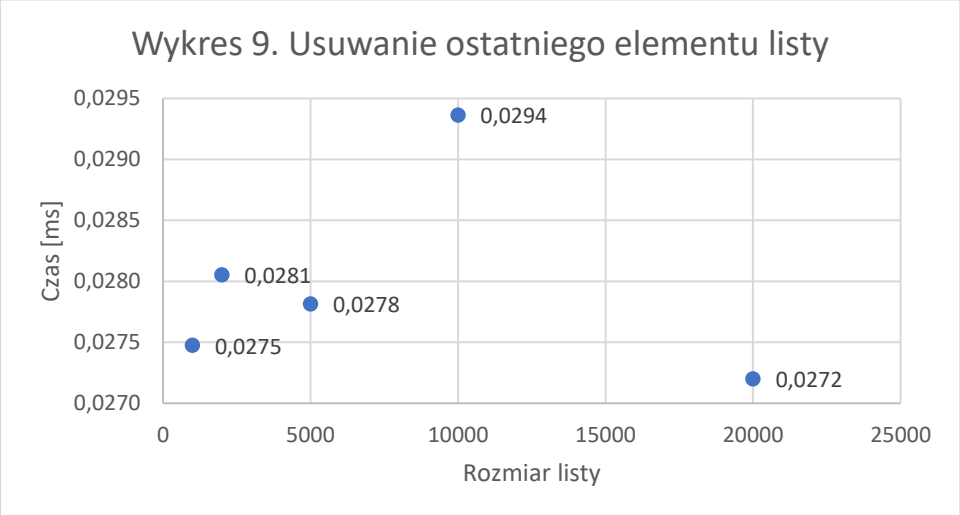
OPERACJE NA LIŚCIE DWUKIERUNKOWEJ

USUWANIE PIERWSZEGO ELEMENTU LISTY



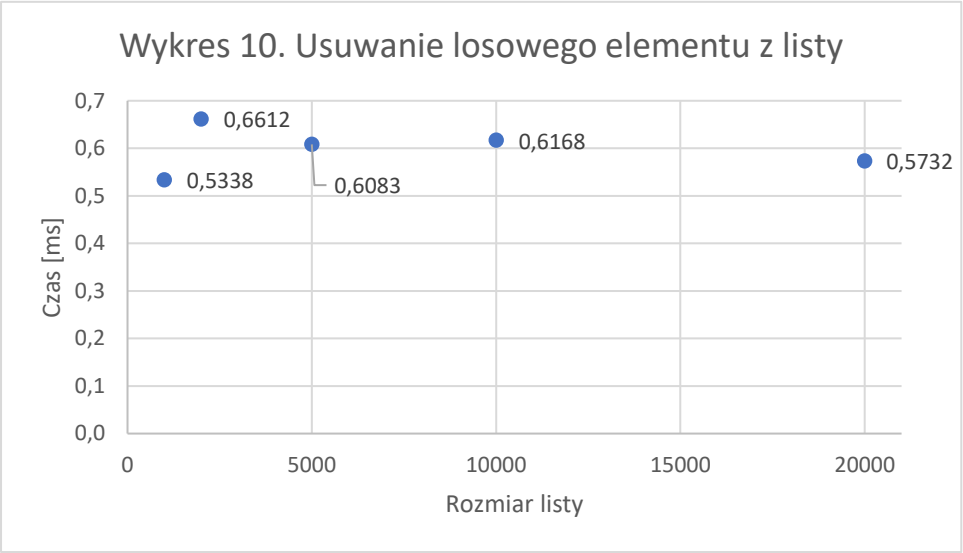
rozmiar	czas[ms]
1000	0,0281
2000	0,0275
5000	0,0270
10000	0,0273
20000	0,0300

USUWANIE OSTATNIEGO ELEMENTU LISTY



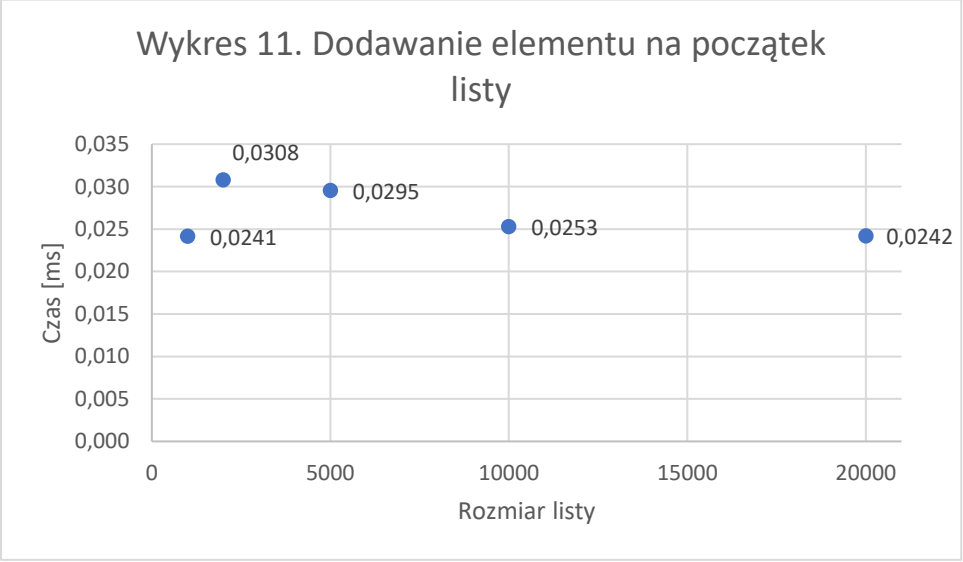
rozmiar	czas[ms]
1000	0,0275
2000	0,0281
5000	0,0278
10000	0,0294
20000	0,0272

USUWANIE LOSOWEGO ELEMENTU LISTY



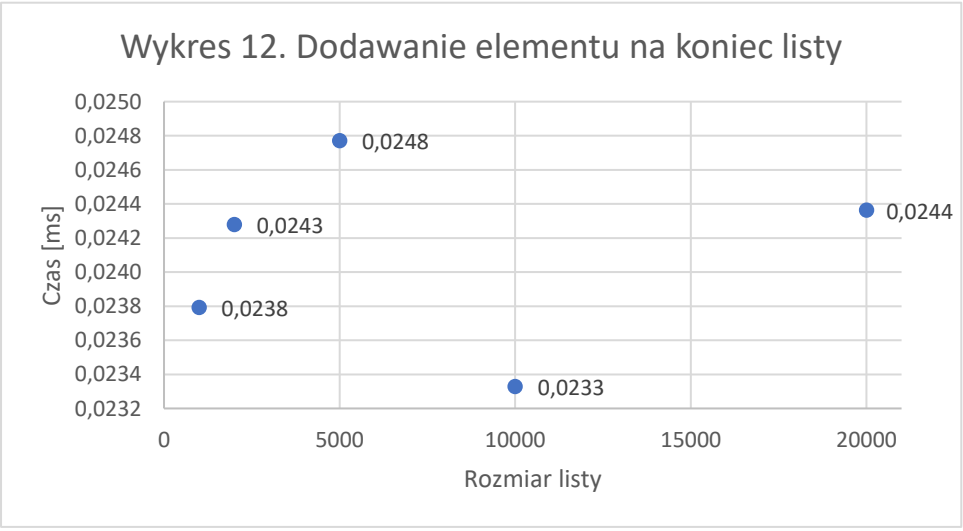
rozmiar	czas[ms]
1000	0,5338
2000	0,6612
5000	0,6083
10000	0,6168
20000	0,5732

DODAWANIE ELEMENTU NA POCZĄTEK LISTY



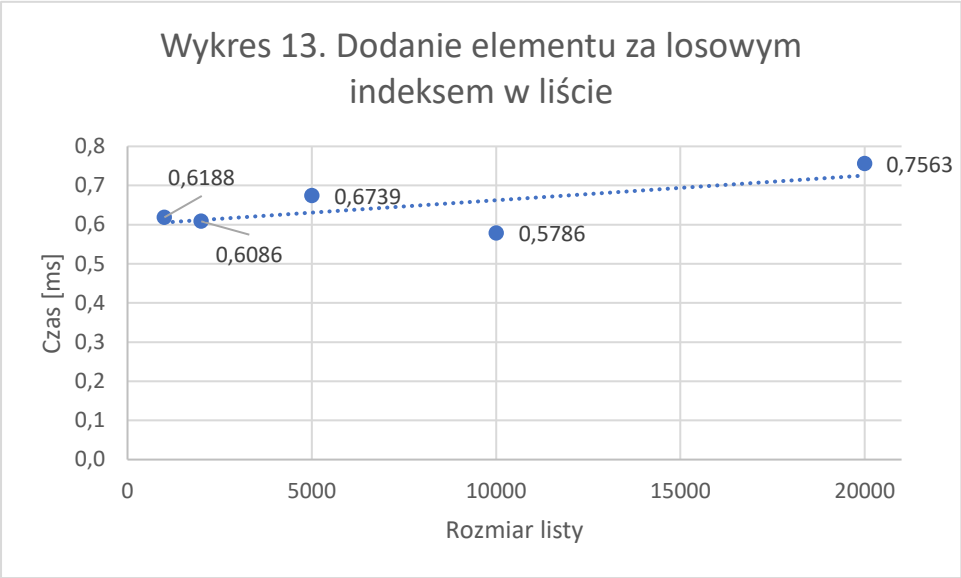
rozmiar	czas[ms]
1000	0,0241
2000	0,0308
5000	0,0295
10000	0,0253
20000	0,0242

DODAWANIE ELEMENTU NA KONIEC LISTY



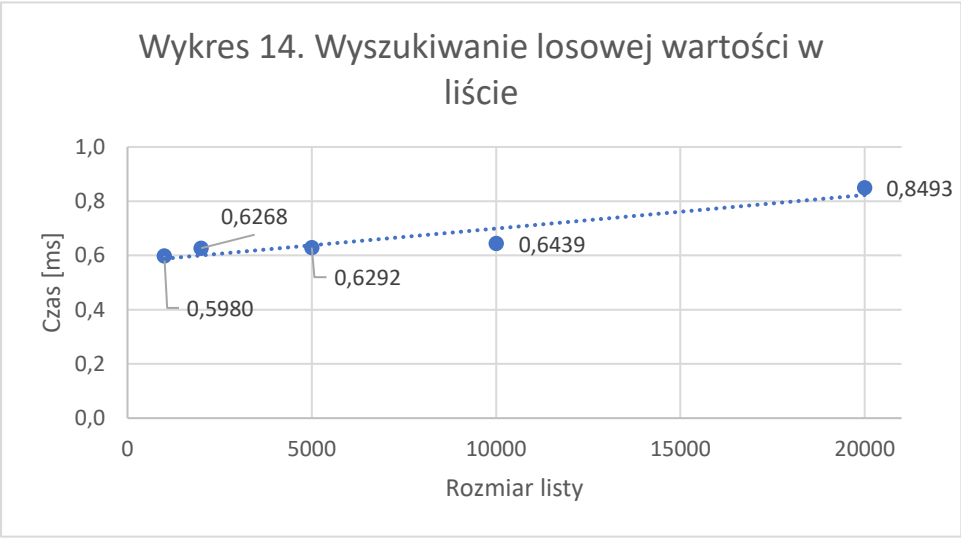
rozmiar	czas[ms]
1000	0,0238
2000	0,0243
5000	0,0248
10000	0,0233
20000	0,0244

DODANIE ELEMENTU ZA LOSOWYM INDEKSEM W LIŚCIE



rozmiar	czas[ms]
1000	0,6188
2000	0,6086
5000	0,6739
10000	0,5786
20000	0,7563

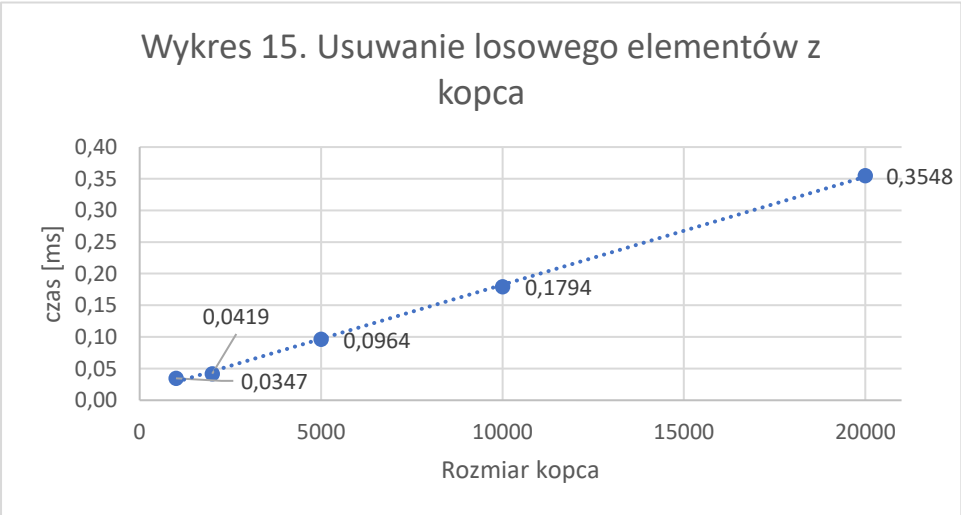
WYSZUKIWANIE ELEMENTU W LIŚCIE



rozmiar	czas[ms]
1000	0,5980
2000	0,6268
5000	0,6292
10000	0,6439
20000	0,8493

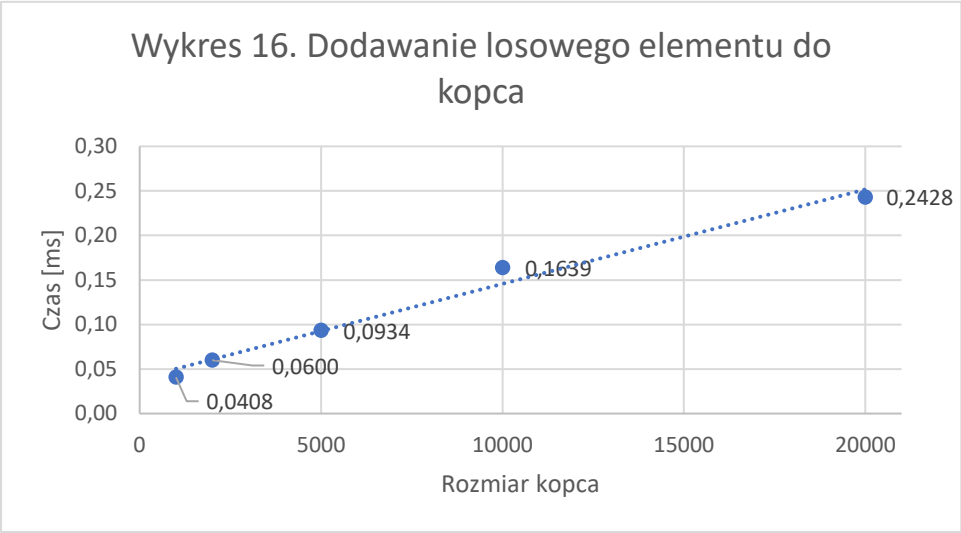
OPERACJE NA KOPCU

USUWANIE ELEMENTÓW Z KOPCA



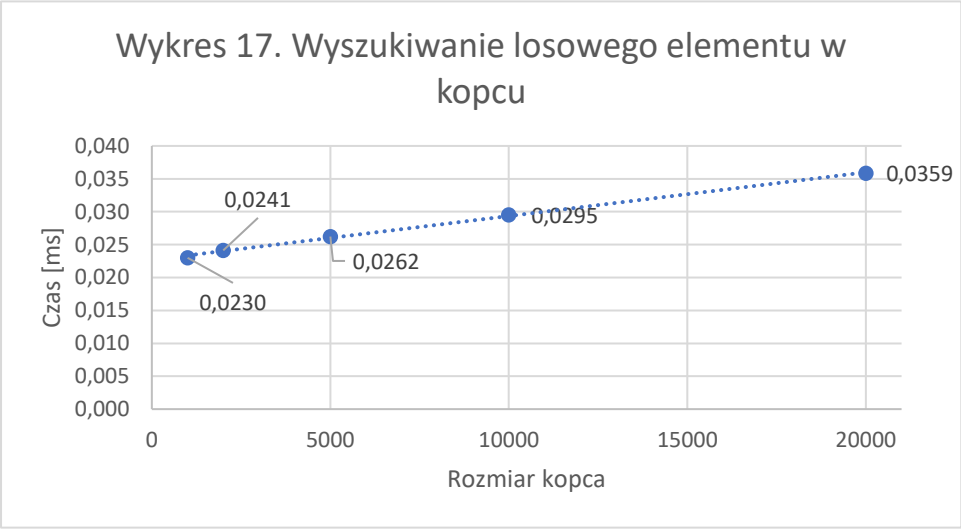
rozmiar	czas[ms]
1000	0,0347
2000	0,0419
5000	0,0964
10000	0,1794
20000	0,3548

DODAWANIE ELEMENTÓW DO KOPCA



rozmiar	czas[ms]
1000	0,0408
2000	0,0600
5000	0,0934
10000	0,1639
20000	0,2428

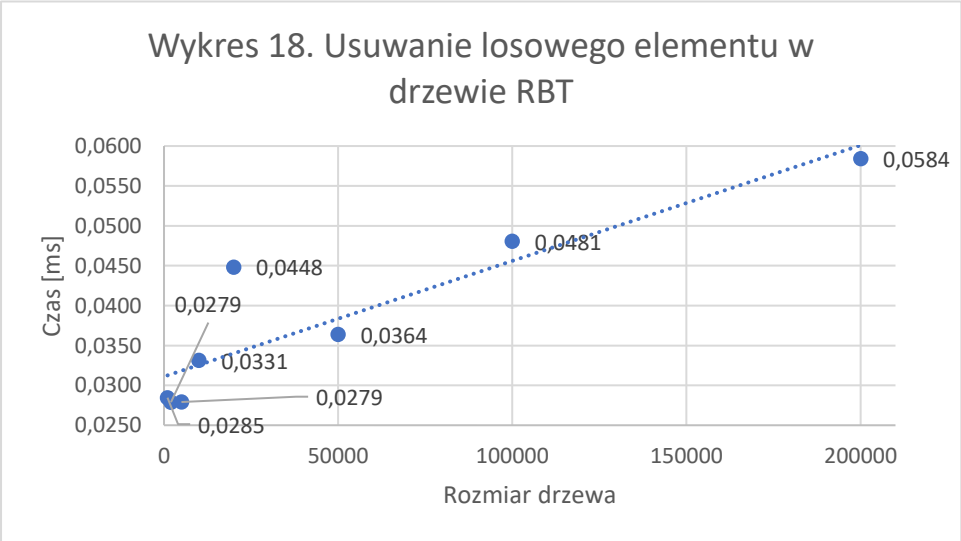
WYSZUKIWANIE ELEMENTÓW W KOPCU



rozmiar	czas[ms]
1000	0,0230
2000	0,0241
5000	0,0262
10000	0,0295
20000	0,0359

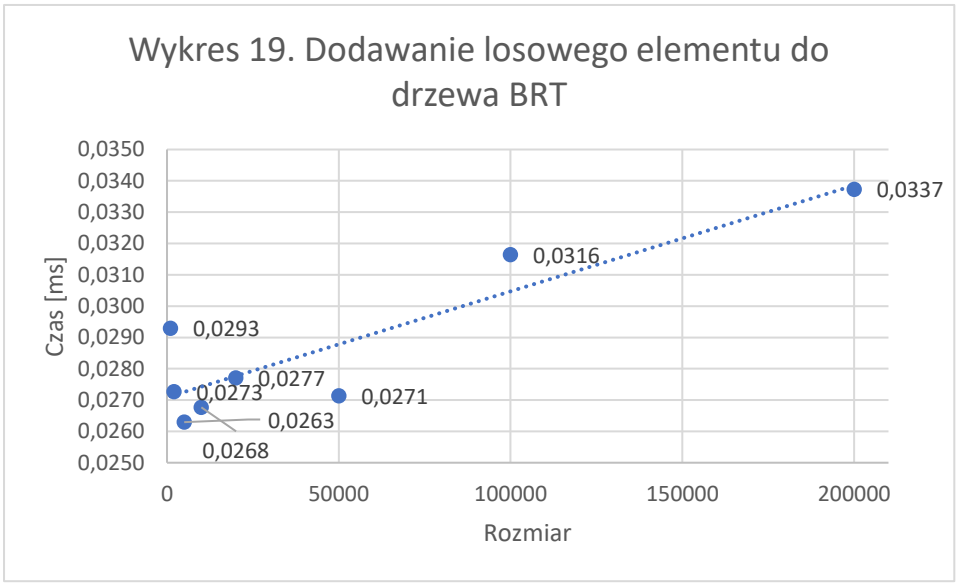
OPERACJE NA DRZEWIE RBT

USUWANIE ELEMENTÓW Z DRZEWA



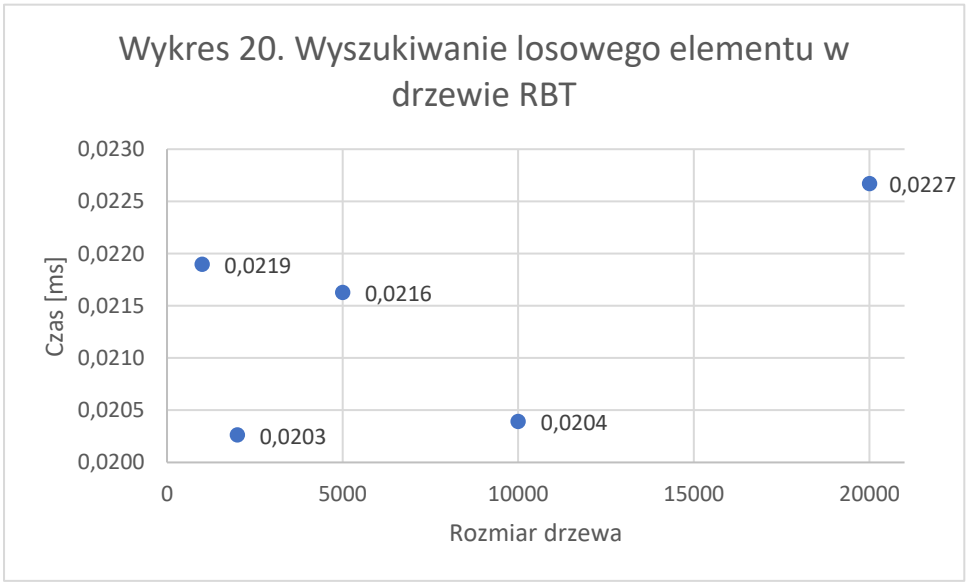
rozmiar	czas [ms]
1000	0,0285
2000	0,0279
5000	0,0279
10000	0,0331
20000	0,0448
50000	0,0364
100000	0,0481
200000	0,0584

DODAWANIE ELEMENTÓW DO DRZEWA



rozmiar	czas[ms]
1000	0,0293
2000	0,0273
5000	0,0263
10000	0,0268
20000	0,0277
50000	0,0271
100000	0,0316
200000	0,0337

WYSZUKIWANIE ELEMENTU W DRZEWIE



rozmiar	czas[ms]
1000	0,0219
2000	0,0203
5000	0,0216
10000	0,0204
20000	0,0227

WNIOSKI

Maksymalny zakres wartości losowanego elementu nieznacznie wydłużył czas operacji, ale nie miał wpływu na samo działanie algorytmu. W przypadku kopca i drzewa zależnie od wartości dodanego elementu, może być konieczne przywrócenie porządku struktury. Dynamiczna alokacja pamięci w przypadku struktury tablicowej jest istotna w prawidłowym działaniu programu, jednak nieprawidłowe jej zaimplementowanie skutkuje niestabilnością programu, spowodowana najczęściej wyciekiem pamięci.

- Operacje na tablicy:

  Wszystkie operacje na tablicy są wprost proporcjonalne do jej rozmiaru. Główny wpływ na ujednolicenie wyników ma wymuszona relokacja po każdej operacji. Gdyby pominąć relokację, czasy usuwania i dodawania z końca tablicy wykonywały by się ze stałym czasem. Natomiast czasy dodawani i usuwania z wnętrza tablicy zleżałyby od indeksu w którym należy wykonać operację – im dalej, tym dłużej. Ze względu na przyjęty sposób odmierzania czasu oraz relokacji tablicy, wyniki pomiarów posiadają znaczny błąd pomiarowy.

- Operacje na liście:

  Dzięki wskaźnikom na głowę i ogon listy operacje na końcu i początku listy (dodawanie i usuwanie z końca lub początku) mają stałe czasy. Czas usunięcia lub wstawienia elementu do wnętrza listy uzależniony jest od położenia modyfikowanego elementu na liście. Podobnie jak w tablicy, średnie czasy wykonywania reszty operacji rosną wraz z rozmiarem listy.

- Operacje na kopcu i drzewie

  Czas trwania algorytmów używanych w kopcu również wzrastają proporcjonalnie do jego rozmiaru oraz wszystkie operacje na drzewach czerwono – czarnych. Jednak wykonując operację na drzewie znacznie zwiększono zakres badanych struktur, ponieważ algorytmy działały zbyt szybko i nie można było zaobserwować dokładnych wyników.

Badane w projekcie struktury mają różne zastosowania. Tablice umożliwiają bezpośredni dostęp do indeksu, jednak przetwarzanie danych trwa znacznie dłużej niż w przypadku listy, która dzięki strukturze wskaźników świetnie sprawdza się w kolejkach i stosach. Kopiec jest dobrą strukturą do sortowania danych i szybkiego ich wyszukiwania. Natomiast największą zaletą drzew czerwono - czarnych jest zrównoważona oraz uporządkowana struktura, pozwalająca na szybką modyfikację elementów.

BIBLIOGIRAFIA

1. *Algorytmy, struktury danych*, autor: mgr Jerzy Wałaszek, www: http://eduinf.waw.pl/inf/alg/001_search
2. Drzewo czerwono-czarne, Wikipedia, www: https://pl.wikipedia.org/wiki/Drzewo_czerwono-czarne
3. Kopiec (binarny) autor: Michał Karpiński, www: http://informatyka.wroc.pl/node/433