Sprawozdanie z pierwszego zadania projektowego z przedmiotu „Struktury danych i Złożoność obliczeniowa”

Michał Zajdel, Nr. Indeksu: 263932

Grupa projektowa: INEK00026P

Kod grupy: K03-37h

Prowadzący: Dr. Inż. Dariusz Banasiak

Spis treści

1. Wstęp 3
Opis eksperymentu 3
2. Tablica 3
Opis 3
Opis operacji 4
Wyniki 5
Wykresy 7
Wnioski 8
3. Lista dwukierunkowa 8
Opis 8
Opis operacji 9
Wyniki 10
Wykresy 13
Wnioski 14
4. Kopiec typu maksimum 14
Opis 14
Opis operacji 15
Wyniki 16
Wykresy 18
Wnioski 19
5. Drzewo Czerwono-Czarne 20
Opis 20
Opis operacji 21
Wyniki 23
Wykresy 25
Wnioski 25

1. Wstęp

Poniższe opisy struktur danych, oraz eksperymentów wykonanych na tych strukturach danych, zostały napisane na podstawie programu, który został załączony do folderu. Kod źródłowy programu znajduje się w folderze „Kod źródłowy”. Projekt został napisany w IDE CLion od firmy JetBrains. W folderze znajduje się także arkusz kalkulacyjny o nazwie „Testy czasowe”, w którym znajdują się wszystkie pomiary zrobione na rzecz projektu. W sprawozdaniu przedstawiłem tylko 20 pierwszych pomiarów czasu, w celu zredukowania miejsca, które zajmują tabele przedstawiające te pomiary czasu. W tabelach przedstawiających średnie pomiary czasu zostały wzięte pod uwagę wszystkie wykonane pomiary dla danej operacji.

Opis eksperymentu

Dla każdej z poniższych struktur danych zmierzono czas zajmujący na poszczególną operację dla następujących wielkości struktur:

- 10000 elementów
- 100000 elementów
- 250000 elementów
- 1000000 elementów

Przy pomiarze czasu nie zostało wzięte pod uwagę generowanie struktur o poszczególnych ilościach elementów. Przy generowaniu tych struktur użyto liczb losowych, wygenerowanych przy pomocy funkcji std::random_device oraz std::uniform_int_distribution. Aby wykonać pomiar czasu użyto funkcji QueryPerformanceCounter.

Kod źródłowy projektu, plik wykonywalny, PDF sprawozdania oraz arkusz .xlsb zawierający pomiary można znaleźć na repozytorium na GitHub:

https://github.com/Huntarman/DataStructuresImplementation

2. Tablica

Opis

Tablica z dynamicznie alokowaną pamięcią , jest strukturą danych pozwalającą na szybkie dodanie oraz usuwanie danych na indeksie końcowym. W przeciwieństwie do tablicy statycznej, jej wielkość jest na bieżąco modyfikowana wraz ze wzrostem ilości danych, które przechowuje w danym momencie. Do podstawowych operacji w tablicy dynamicznej należą:

- Dodawanie elementu
  - Na końcu (indeksie końcowym)
    - Złożoność obliczeniowa : Θ(1)
  - Na początku (indeksie 0)
    - Złożoność obliczeniowa : Θ(n)
  - W środku tablicy
    - Złożoność obliczeniowa : Θ(n-i), gdzie i to indeks elementu dodawanego, rzeczywista złożoność obliczeniowa : Θ(n)
- Usuwanie elementu
  - Na końcu (indeksie końcowym)
    - Złożoność obliczeniowa : Θ(1)
  - Na początku (indeksie 0)
    - Złożoność obliczeniowa : Θ(n)

    - W środku tablicy
        - Złożoność obliczeniowa : Θ(n-i), gdzie i to indeks elementu usuwanego, rzeczywista złożoność obliczeniowa : Θ(n)
- Wyszukanie elementu
    - złożoność obliczeniowa : Θ(i), gdzie i to indeks elementu wyszukiwanego, rzeczywista złożoność obliczeniowa : Θ(n)

Opis operacji

Dodawanie elementu na końcu tablicy jest wykonywana poprzez realokację jej wielkości oraz wstawienie nowego elementu na indeks n-1 gdzie n to ilość elementów w tablicy (wliczając element dodawany). Jednak w przypadku dodawania na początku tablicy, lub na indeksie środkowym wykonywane jest „przesunięcie" wartości o jeden indeks do przodu. W przypadku dodawania elementu na początku należy zrealokować wielkość tablicy, i iterując od indeksu n-1 (znowu licząc element dodawany), przypisywać kolejnym miejscom w tablicy wartość elementu o indeksie mniejszym o 1. W przypadku dodawania elementu na indeksie środkowym, należy działać w sposób analogiczny, jednak nie do początku tablicy, a do indeksu na który ma być wstawiony nowy element.

Przy usuwaniu elementu na końcu tablicy możemy posłużyć się realokacją pamięci, po zmniejszeniu zmiennej zawierającej ilość elementów w tablicy. W przypadku usuwania na początku lub w indeksie środkowym, należy przesunąć odpowiednie elementy na indeks o 1 mniejszy, po czym zrealokować pamięć przeznaczoną na przechowanie tablicy.

Wyszukanie elementu zachodzi przez porównywanie elementów tablicy z poszukiwanym elementem, zaczynając od początku tablicy. Jeśli element zostanie znaleziony, operacja kończy się sukcesem i porównywanie ustaje. Jeśli operacja porówna wszystkie elementy tablicy, ale nie znajdzie poszukiwanego elementu, operacja zakończy się porażką.

Wyniki

Poniżej znajdują się tabele przedstawiające zmierzone czasy wykonywania operacji na tablicy, oraz tabele przedstawiającą średnie czasów wykonania operacji.

Wstawianie											
Koniec				Początek (indeks 0)				Indeks środkowy			
Wielkość tablicy				Wielkość tablicy				Wielkość tablicy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych											
1 - 100000											
Czas w mikrosekundach				Czas w mikrosekundach				Czas w mikrosekundach			
1,2	0,6	0,6	1,6	22,05	174,8	530,5	1920	7,8	69,8	170,2	760,5
0,3	0,6	0,3	0,9	34,8	140,3	566,2	2927,9	7,4	69,5	174,8	735,1
0,4	0,2	0,2	1	21,02	139,5	517,9	2584,7	6,7	69,3	178,6	730
0,2	0,2	0,2	1,1	21,03	139,4	513,2	1917,4	6,5	70,8	177	696,2
0,2	0,1	0,1	1	41,3	139,5	605	1918,1	8,4	71	177,9	722,4
0,1	0,1	0,1	1,4	22,05	139,5	575	1933,5	7,1	69,4	178,3	689,5
0,1	0,2	0,1	0,8	24,01	139,4	522,4	1862,3	8,5	69,3	182,6	697,1
0,1	0,1	0,1	1,2	30,06	139,4	518,1	1893	6,4	69,4	174,3	751,8
0,1	0,1	0,1	1,6	21,02	139,5	663	1947	7,1	69,3	172,8	709,5
0,1	0,2	0,2	1,5	21,03	145,2	790,9	2166,1	7	71,2	175,2	690,1
0,1	0,1	0,1	1,6	26,05	139,8	548,2	2020,1	7,1	68,7	168,9	694,4
0,1	0,1	0,4	1,5	22,02	136,3	514,8	1869,7	7	69,2	169,1	751,6
0,1	0,2	0,1	1,3	21,08	139,9	529,5	2003,8	8,5	69,3	169,3	755,9
0,2	0,2	0,1	0,7	25,07	136,4	586,5	2946,1	7	69,2	169,1	700,3
0,2	0,1	0,1	1,1	22,02	149,9	663,7	1894,7	7,1	69,3	169,6	718,2
0,1	0,1	0,1	1,5	21	138,2	507,6	2032,9	7	68,7	178,8	694,5
0,2	0,6	0,8	1,1	24,06	186,9	504,8	2081,9	7	69,2	169,3	717
0,2	0,1	0,3	2	29,04	143,7	731,8	2000,6	7,1	67,1	169,3	770,6
0,2	0,6	0,2	1,5	21,05	146,8	496,8	1993,6	7,1	67,7	169,6	709,5
0,2	0,1	0,4	1,9	32,7	142,7	846,3	2202,3	7,4	70,7	169,2	842,7

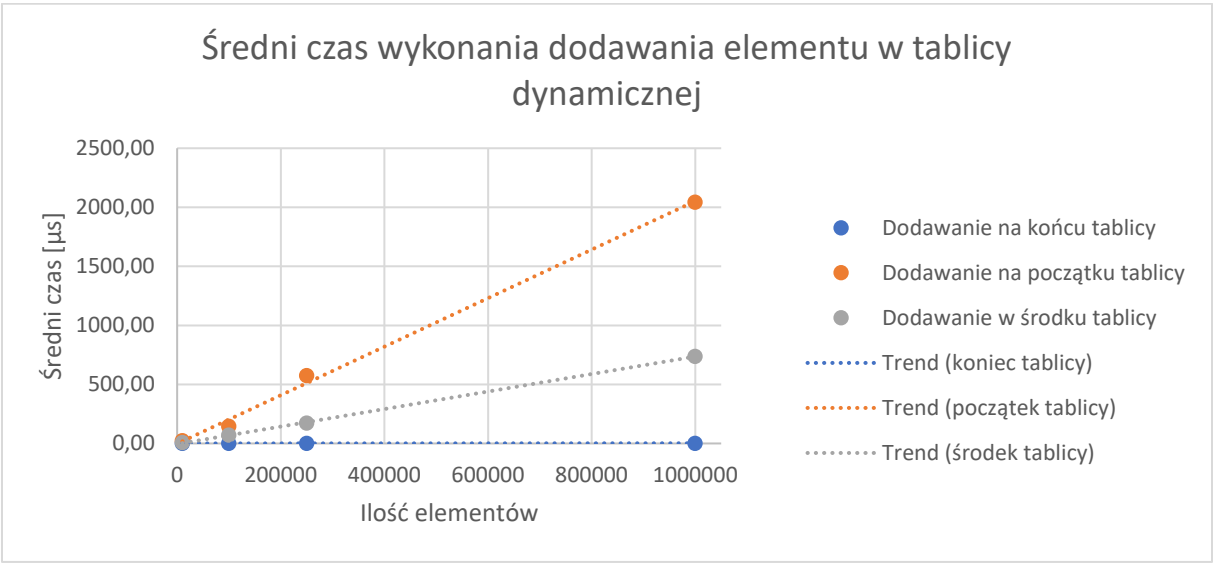
Usuwanie											
Koniec				Początek (indeks 0)				Indeks środkowy			
Wielkość tablicy				Wielkość tablicy				Wielkość tablicy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych											
1 - 100000											
Czas w mikrosekundach				Czas w mikrosekundach				Czas w mikrosekundach			
0,4	0,3	0,5	1,2	33,7	174,3	518,8	1993,9	7,7	71,5	173,4	694
0,3	0	0,2	3,5	22,3	138,6	502,3	2307	7,3	71,4	174,4	711,1
0,1	0,1	0,1	1,9	29	136,5	500,5	1916,1	6,9	73,5	169,5	700,8
0,1	0,1	0,1	0,9	25,8	136,4	560,3	1874,7	7	71,1	169,6	757,9
0,1	0	0,1	1,7	23,9	136,4	508,7	2159,8	7	70,9	178,2	790,3
0,1	0,1	0,1	1,7	23,7	136,3	572,6	1903,3	7	71	177,3	701,1
0,1	0,1	0,3	6	26	136,5	587,9	1947,1	10,6	71,1	173,3	767,5
0,1	0,1	0,2	2,2	29,3	141,2	637,6	1853	7,3	71,1	177,2	713,1
0,1	0,3	0,1	1,5	24,6	145,1	485	1932,1	7,2	70,9	169,4	710,1
0,1	0,1	0,1	0,9	29,9	136,3	1059,9	2661,5	7	71,1	169,5	709,1
0	0	0	1,2	21,3	136,3	530,3	1832,5	6,9	69,5	173,6	716
0	0	0,1	1,4	33,2	136,2	549,6	1888,7	7	69,4	169,5	713,9
0	0,1	0,1	1,2	28,2	136,2	523,5	1925,6	6,9	69,3	174,2	689,1
0,1	0,1	0,1	1,5	21,5	138,4	522,5	1823,8	7	69,5	178,7	749,7
0,1	0,1	0,1	1	22,3	150,6	554	2190,1	7	69,5	169,7	831,1
0	0,1	0	0,5	32	140,3	603,7	1959,9	7,1	69,5	169,6	736,3
0,1	0,1	0,2	0,9	20,8	139,1	550,1	1832,1	6,9	69,5	169,4	742,7
0,1	0,1	0,2	0,7	25,3	159	500,4	1891,7	7	69,4	169,8	766,9
0,1	0,1	0,1	0,6	29,5	145,9	554,9	1832,1	7	69,6	169,3	738,1
0,1	0	0,1	0,6	21,1	142,2	571,4	2372,3	7	69,4	170	791,9

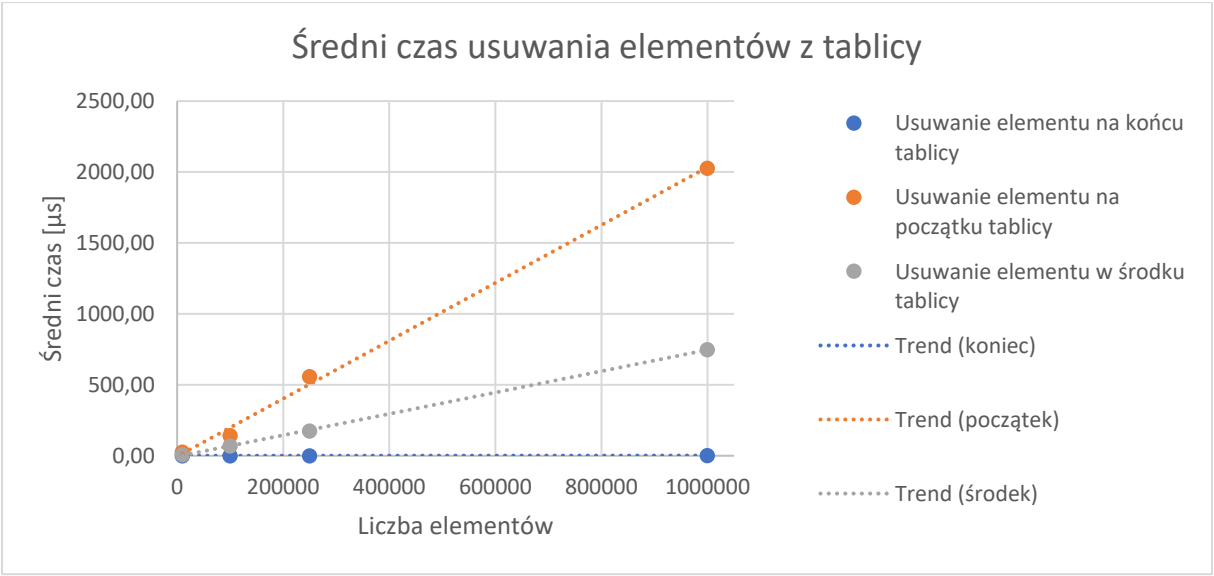
Szukanie							
Losowa wartość							
Wielkość tablicy							
10000		100000		250000		1000000	
Zasięg liczb losowych							
1 - 100000				1 - 1000000			
Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?
10,9	nie	9,8	tak	266,7	nie	1103,3	nie
10,2	nie	76,8	tak	229,6	tak	66,4	tak
10,1	nie	33,1	tak	267,7	nie	130,9	tak
10,2	nie	34,4	tak	253,1	nie	623,9	tak
10,2	nie	101,1	nie	284,6	nie	269,5	tak
10,2	nie	74,4	tak	265,2	nie	609,7	tak
10,2	nie	103,4	nie	264,7	nie	1249,7	nie
10,1	nie	67,7	tak	254,1	nie	322,7	tak
10,1	nie	43	tak	262	nie	210,6	tak
9,2	tak	86,4	tak	252,8	nie	917,8	tak
10,2	nie	1,6	tak	253	nie	214,7	tak
10,1	nie	38,7	tak	261,7	nie	1043,7	nie
10,2	nie	29,2	tak	24,7	tak	82,3	tak
1,4	tak	20,2	tak	81,5	tak	1095,7	nie
10,1	nie	101,2	nie	277,2	nie	59,3	tak
10,1	nie	101,2	nie	198,5	tak	979,9	tak
10,1	nie	107	nie	256,6	nie	574,3	tak
10,9	nie	94,1	tak	252,8	nie	166,6	tak
10,2	nie	101,4	nie	252,7	nie	1058,6	nie
9,7	tak	14,2	tak	268,7	nie	1096,6	nie

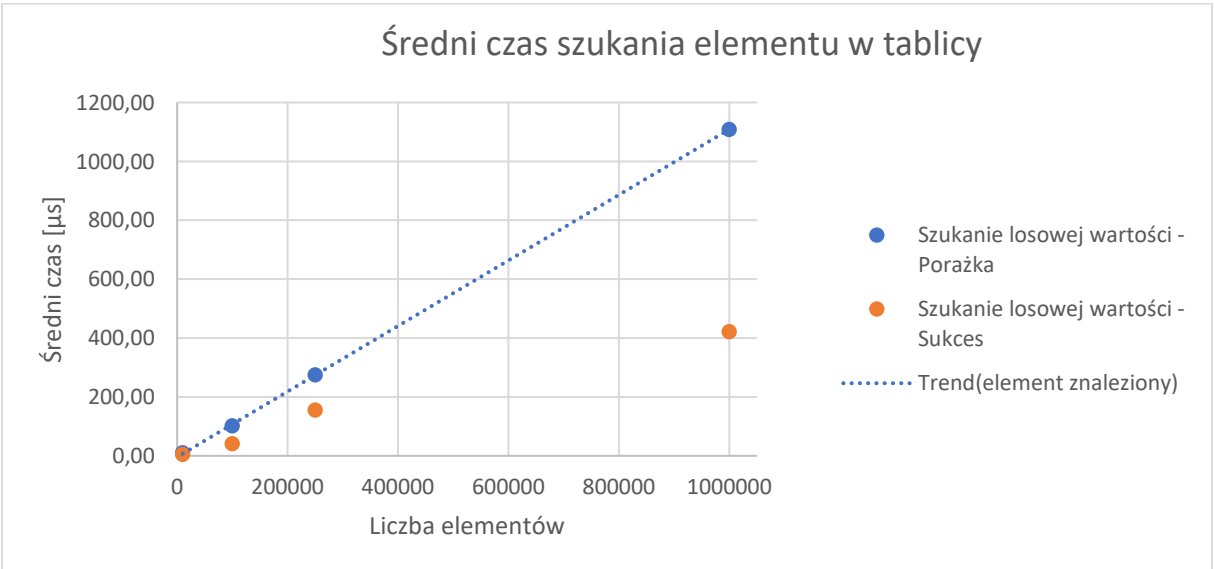
Średnie pomiarów czasu

Wstawianie											
Koniec				Początek (indeks 0)				Indeks środkowy			
Wielkość tablicy				Wielkość tablicy				Wielkość tablicy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,17	0,29	0,26	1,45	24,21	144,13	574,29	2042,60	7,14	69,84	173,00	738,06
Usuwanie											
Koniec				Początek (indeks 0)				Indeks środkowy			
Wielkość tablicy				Wielkość tablicy				Wielkość tablicy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,07	0,20	0,14	1,33	25,48	142,52	557,36	2025,88	7,12	70,09	174,93	748,10
Szukanie											
Losowa wartość											
Wielkość tablicy											
10000	100000	250000	1000000	10000	100000	250000	1000000				
Porażka				Sukces							
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)							
10,59	102,30	275,04	1108,88	4,76	41,18	155,47	421,67				

Wykresy


Średni czas wykonania dodawania elementu w tablicy dynamicznej
Średni czas [µs]
2500,00
2000,00
1500,00
1000,00
500,00
0,00
0
200000
400000
600000
800000
1000000
Ilość elementów
Dodawanie na końcu tablicy
Dodawanie na początku tablicy
Dodawanie w środku tablicy
Trend (koniec tablicy)
Trend (początek tablicy)
Trend (środek tablicy)


Średni czas usuwania elementów z tablicy
Średni czas [µs]
2500,00
2000,00
1500,00
1000,00
500,00
0,00
0
200000
400000
600000
800000
1000000
Liczba elementów
Usuwanie elementu na końcu tablicy
Usuwanie elementu na początku tablicy
Usuwanie elementu w środku tablicy
Trend (koniec)
Trend (początek)
Trend (środek)


Średni czas szukania elementu w tablicy
Średni czas [µs]
1200,00
1000,00
800,00
600,00
400,00
200,00
0,00
0
200000
400000
600000
800000
1000000
Liczba elementów
Szukanie losowej wartości - Porażka
Szukanie losowej wartości - Sukces
Trend(element znaleziony)

Linie trendu zostały przedstawione dla operacji których złożoność czasu wykonywania da się jednoznacznie określić. W przypadku dodawania/usuwania elementu na końcu tablicy, operacja wykonywana jest niemalże natychmiastowo. W przypadku: dodawania/odejmowania w środku/na początku tablicy oraz porażki przy szukaniu elementu, czas wykonywania operacji rośnie liniowo.

Trudno jednoznacznie określić rzeczywisty trend funkcji wyszukiwania, gdy zakończy się ona sukcesem. Jeśli element wyszukiwany znajdzie się na indeksie 0, zakończy się ona natychmiastowo; jeśli znajdzie się na ostatnim miejscu w tablicy, funkcja zakończy się dopiero po przejściu przez wszystkie inne elementy.

Wnioski

Tablica jest strukturą danych, której zaletą jest złożoność obliczeniowa stała dodawania i usuwania elementów na swoim końcu. Niestety mutacja w środku lub wyszukanie elementu są złożoności liniowej, w czasie gdy istnieją struktury danych w których takie operacje mają niższą złożoność.

Oznacza to że tablica dynamiczna jest odpowiednią strukturą do przechowania danych, których kolejność nie ma znaczenia, co oznacza możliwość dodawania elementu zawsze na końcu, oraz gdy nie są nam potrzebne szczególne elementy, których znalezienie potrzebowałoby przeszukania tablicy.

3. Lista dwukierunkowa

Opis

Lista dwukierunkowa to struktura danych, której każdy element zawiera, poza wartością, wskaźnik na poprzedni oraz następny element listy. Należy założyć że w liście zapisane są zmienne zawierające wskaźniki jej początkowy element (tył – wizualizując listę, element najbardziej na lewo) oraz końcowy element (przód – element najbardziej na prawo) . Każdy element listy wymaga więcej zarezerwowanej pamięci niż element w tablicy, zawierający wyłącznie wartość. Lista jest strukturą, przypominającą tablicę, jednak w przypadku tablicy na wskazanie określonych elementów używane są indeksy, a w liście – kolejne wskaźniki na następne wartości zaczynając od tyłu listy.

Do podstawowych operacji na liście należą:

- Dodawanie elementu
    - Na końcu (indeksie końcowym)
        - Złożoność obliczeniowa : Θ(1)
    - Na początku (indeksie 0)
        - Złożoność obliczeniowa : Θ(1)
    - W środku tablicy
        - Złożoność obliczeniowa : Θ(i) + Θ(1), gdzie i to „indeks” elementu dodawanego do listy licząc od tyłu listy, rzeczywista złożoność obliczeniowa : Θ(n)

- Usuwanie elementu
  - Na końcu (przodzie listy)
    - Złożoność obliczeniowa : Θ(1)
  - Na początku (tyle listy)
    - Złożoność obliczeniowa : Θ(1)
  - W środku listy
    - Złożoność obliczeniowa : Θ(i) + Θ(1), gdzie i to „indeks” elementu usuwanego z listy licząc od tyłu listy, rzeczywista złożoność obliczeniowa : Θ(n)
- Wyszukanie elementu
  - złożoność obliczeniowa : Θ(i), gdzie i to indeks elementu wyszukiwanego, rzeczywista złożoność obliczeniowa : Θ(n)

Opis operacji

Lista, przy deklaracji zawiera wskaźniki na tył oraz przód listy, jednak przed dodaniem elementu oba wskaźniki wskazują na wartość NULL. Dopiero przy dodaniu elementu zostają one zmienione na wskaźnik na ten element. Potem są zmieniane, w zależności dodawania/usuwania elementu na przód bądź tył.

Zadeklarowany element listy posiada określoną wartość, oraz wskaźniki na poprzedni i następny element o początkowej wartości NULL. Jednak w trakcie operacji, te wskaźniki są nadpisywane, aby wskazywać na rzeczywiste elementy listy. Jeśli po wykonaniu operacji wskaźnik na poprzedni element jest NULL’em, oznacza to że element jest na tyle listy. Jeśli po wykonaniu operacji wskaźnik na następny element jest NULL’em, oznacza to że element jest na przodzie listy.

Przy znanych wskaźnikach na przód i tył listy, złożoność dodawania lub usuwania elementu na przód czy tył listy jest stała. Przy mutacji listy na indeksie środkowym liczba wykonanych akcji często będzie mniejsza niż w przypadku takiej samej operacji w tablicy dynamicznej, jednak konieczność przypisywania do wskaźnika na aktualną element adresu na następne elementy, zwiększy realny czas potrzebny na taką operację. Szukanie elementu będzie wymagało tyle samo porównań co w tablicy, jednak konieczność przypisywania do wskaźnika adresu na kolejny element, sprawia że wyszukanie elementu realnie zajmuje więcej czasu niż w tablicy.

Dodawanie elementu na przód listy to stworzenie elementu o podanej wartości, ustawienie jego wskaźnika na poprzedni element na adres aktualnego przodu listy, ustawienie wskaźnika na następną wartość aktualnego końca listy na adres elementu dodawany i w końcu ustawienie wskaźnika na przód listy na adres dodawanego elementu. Dodawanie elementu na tył listy to jest wykonane analogicznie, oczywiście biorąc pod uwagę zamianę innych adresów (przypisanie wskaźnika na następny element, zamiast na poprzedni etc.). Usuwanie elementów z przodu i z tylu zachodzi poprzez ustawienie wskaźnika na wartość poprzednią bądź kolejną kolejnego/poprzedniego elementu aktualnego przodu/tyłu na NULL, ustawienia wskaźnika na przód/tył oraz usunięcie dotychczasowego przodu/tyłu.

Sytuacja komplikuje się przy dodawaniu/usuwaniu elementu w środku listy. Wtedy zaczynając od wskaźnika na przód tablicy, przechodzimy przez wskaźniki na kolejne elementy, aż nie trzymamy wskaźnika na element, przed którym nowy element zostanie dodany/usunięty. W przypadku dodawanie ustawiamy odpowiednie wskaźniki żeby nowy element miał przypisane poprawne wskaźniki i zamieniamy także odpowiednie wskaźniki elementów: poprzedniego i następnego. Wybieranie elementu następuje identycznie dla usuwania elementu w środku listy,

jednak po osiągnięciu elementu usuwanego zachodzi poprzez przypisanie odpowiednich adresów wskaźnikom poprzedniego i następnego elementu, aby nie wskazywały już na element usuwany oraz zwolnienie pamięci na adresie elementu usuwanego.

Szukanie elementu działa podobnie jak szukanie elementu w tablicy, jednak przechodzimy nie przez kolejne indeksy a przez kolejne wskaźniki na elementy, tak długo jak wskaźnik na następny element aktualnego elementu nie jest równy NULL.

Wyniki

Poniżej znajdują się tabele przedstawiające zmierzone czasy wykonywania operacji na tablicy, oraz tabele przedstawiającą średnie czasów wykonania operacji.

Wstawianie											
Przód				Tył				Indeks środkowy			
Wielkość listy				Wielkość listy				Wielkość listy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych											
1 - 100000											
Czas w mikrosekundach				Czas w mikrosekundach				Czas w mikrosekundach			
0,2	0,1	0,1	0,1	0,1	0,3	0,2	0,2	17,4	384,4	1325,5	4600,5
0,1	0,1	0,1	0,1	0,1	0,2	0,3	0,1	20,7	267,6	1075,9	4384,8
0,1	0,1	0,1	0,1	0,1	0,2	0,1	0,2	18	263,4	972,7	4451,5
0,1	0,1	0,1	0	0,1	0,2	0,2	0,1	18,3	347,8	1056,3	4815,2
0,1	0,1	0,1	0,1	0,1	0,2	0,3	0,1	22,3	284,5	865,3	4253
0,1	0,1	0,1	0,1	0,1	0,2	0,2	0,1	23	331,3	951,1	4821
0,1	0,1	0,3	0,1	0,1	0,2	0,2	0,1	22,8	366	1013,1	4822,1
0,1	0,1	0,1	0	0,1	0,1	0,2	0,2	22,3	305	988,7	4645,3
0,1	0,1	0,1	0,1	0,1	0,1	0,3	0,2	16,8	325,8	1015	4647
0,1	0,1	0,1	0	0,1	0,1	0,2	0,1	26,3	315,4	978,7	4456,5
0,2	0,1	0,1	0	0,1	0,1	0,2	0,2	21,7	310,2	1277,3	4559,9
0,1	0,1	0,1	0,1	0,1	0,1	0,2	0,2	19,2	296,5	872,3	4657,8
0,1	0	0,1	0	0,1	0,1	0,3	0,2	18,8	264	968,4	4795,4
0,1	0,1	0	0,1	0,1	0,3	0,3	0,1	18,3	264,8	1281,4	4805,7
0,1	0	0,1	0,1	0,1	0,1	0,2	0,2	19,4	257,7	1338,4	5077
0,1	0	0,1	0	0,1	0,1	0,2	0,2	24,9	265,4	1022,3	4299,4
0,1	0,1	0,1	0,1	0,2	0,1	0,2	0,3	21,4	391,7	1111,6	4399,9
0	0,1	0,1	0,1	0,1	0,2	0,1	0,2	20,8	302	1033,1	5041,7
0,1	0,1	0,1	0,1	0,2	0,1	0,1	0,2	19	314,4	943,1	4312,8
0	0,1	0,1	0	0,1	0,2	0,2	0,2	24,2	313,4	1363,9	4816,7

Usuwanie												
Przód				Tył				Indeks środkowy			
Wielkość listy				Wielkość listy				Wielkość listy			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych											
1 - 100000											
Czas w mikrosekundach				Czas w mikrosekundach				Czas w mikrosekundach			
0,4	0,6	0,6	0,5	0,5	0,6	0,7	0,9	24	473	1274,5	4733,3
0,1	0,2	0,4	0,5	0,1	0,3	0,4	0,4	23,4	361,5	1115,1	4915,7
0,1	0	0,5	0,4	0	1,3	0,5	0,9	24	419,5	1063,3	4577,4
0	0,2	0,2	0,5	0,1	0,4	0,3	0,9	23,7	411,4	1184,5	4838,9
0	0,2	0,4	0,4	0	0,3	0,5	0,7	12,4	523,8	1193,4	4660
0,1	0,1	0,2	0,5	0,1	0,2	0,6	0,4	22,9	480,9	1004,6	5004,1
0	0,4	0,2	0,4	0	0,3	0,4	1	23	367,2	930,3	4615,7
0	0,1	0,2	0,5	0,1	0,3	0,4	0,7	19,1	333,3	1107,4	4473,1
0	0,2	0,3	0,7	0,1	0,2	0,3	1,1	22	316,1	1507,2	4735,9
0	0,2	0,3	0,6	0,1	0,4	0,4	0,8	24,1	279,4	1176,8	5045,4
0	0,1	0,1	0,4	0,1	0,2	0,2	0,9	20	303,8	1099,3	4685,6
0,1	0,2	0,4	0,4	0,1	0,3	0,3	0,6	22,7	307,5	792,6	4689
0	0,2	0,2	0,7	0,1	0,2	0,4	0,6	23,3	294,9	990,7	5435,8
0	0,1	0,4	0,3	0	0,2	0,4	0,5	22,7	547,4	1307,5	4772,2
0,1	0	0,3	0,6	0	0,4	0,4	0,8	19,5	296,6	1053,4	5010,6
0	0,1	0,2	0,5	0,2	0,1	0,5	0,7	20,6	259,3	977,4	3977
0,1	0,1	0,2	0,5	0,1	0,1	0,3	0,7	11,8	289,8	1043,8	4882,9
0	0,3	0,3	0,5	0	0,7	0,3	0,6	20,2	238,9	1346,6	5123,4
0	0,1	0,6	0,5	0	0,2	0,3	0,6	23,8	418,5	959,3	5533,6
0	0,1	0,3	0,4	0	0,3	0,2	0,8	24,5	294,6	949,3	4879,4

Szukanie							
Losowa wartość							
Wielkość listy							
10000		100000		250000		1000000	
Zasięg liczb losowych							
1 - 100000				1 - 10000000			
Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?
33,7	nie	33,5	tak	1344,7	tak	336	tak
33,7	nie	520,7	nie	615,2	tak	536,1	tak
36,1	nie	13,1	tak	1076	tak	704,9	tak
34,6	nie	474,8	nie	2535,6	nie	7467,8	nie
33,6	nie	716,2	nie	2392	nie	7969,3	nie
33,4	nie	411,4	tak	1871,8	nie	7547,5	nie
34,3	nie	84,3	tak	294,9	tak	6583,3	nie
35,1	nie	418,7	tak	850,6	tak	652,4	tak
34,7	nie	19,9	tak	1800,2	tak	6022,3	tak
44,9	nie	437,1	nie	841,8	tak	7367,7	nie
33,3	nie	356,9	tak	1735,3	nie	6520,1	tak
34,8	nie	324,4	tak	1464,8	tak	5539,1	tak
26,8	tak	140,3	tak	775	tak	4136	tak
33,9	nie	471,7	nie	1712	nie	7044,6	nie
33,7	nie	354,9	tak	1061,7	tak	7055	nie
33,6	nie	547,2	nie	1729,8	nie	6834,9	nie
30,1	tak	153,9	tak	1882,6	nie	2757,7	tak
34,6	nie	297,4	tak	2065,4	nie	7268,6	nie
34,7	nie	502,8	nie	2532,6	nie	1845,5	tak
35,4	nie	428	nie	1751,4	tak	4801,1	tak

Średnie pomiary czasu

Wstawianie

Przód

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
0,10	0,09	0,12	0,09

Tył

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
0,15	0,19	0,22	0,20

Indeks środkowy

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
21,49	295,15	1067,68	4561,38

Usuwanie

Przód

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
0,06	0,17	0,30	0,39

Tył

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
0,091	0,30	0,43	0,80

Indeks środkowy

Wielkość listy			
10000	100000	250000	1000000
Średni czas (mikrosekundy)			
21,48	316,78	1065,63	4744,87

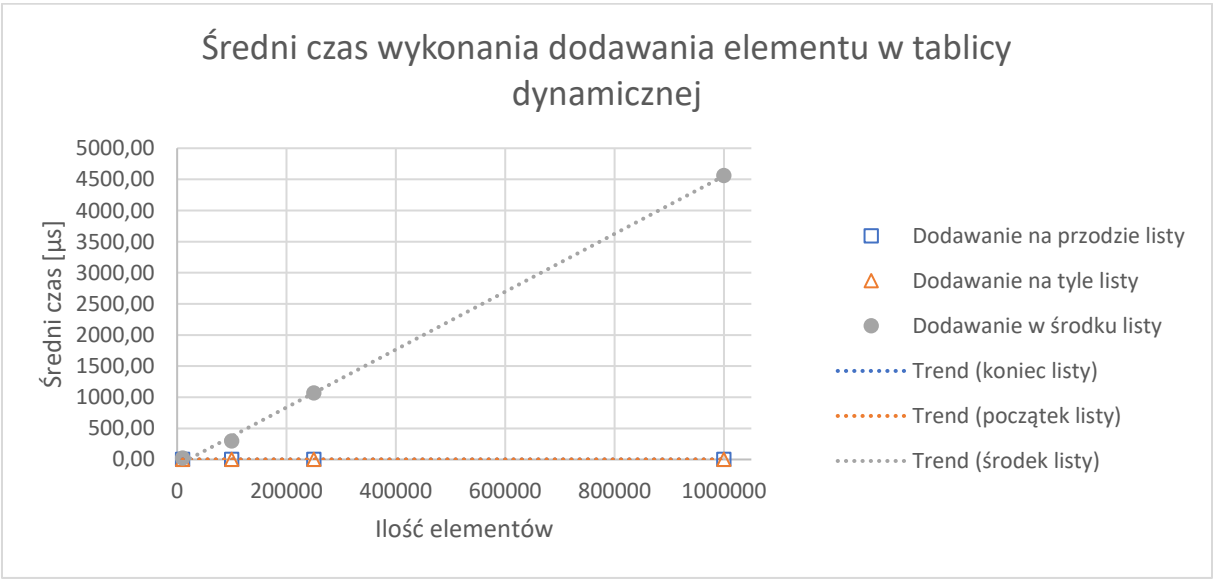
Szukanie

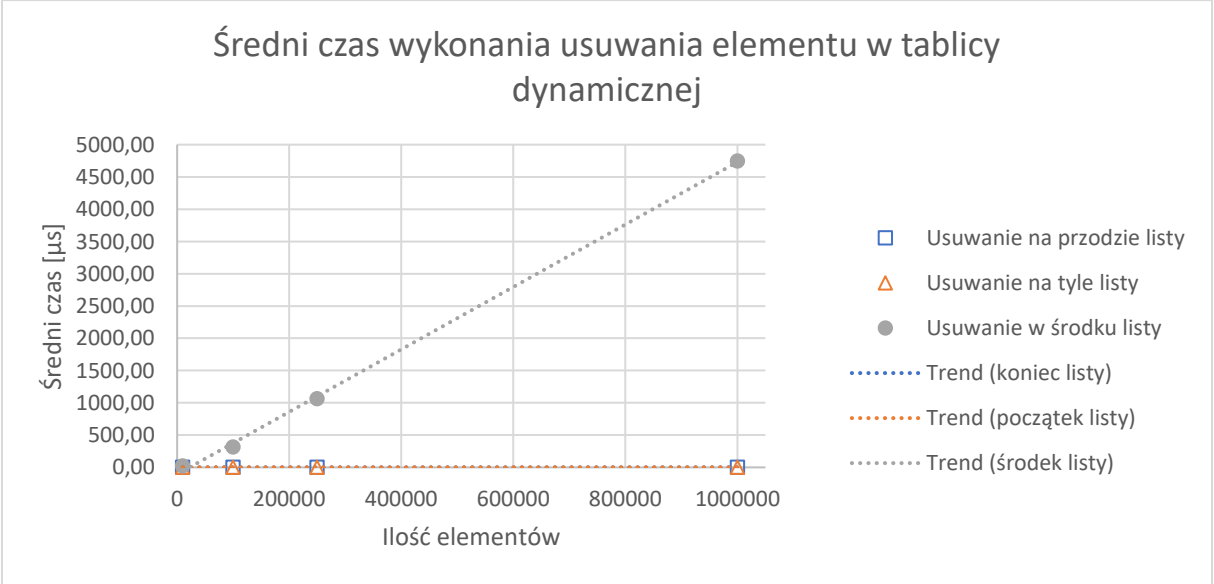
Losowa wartość

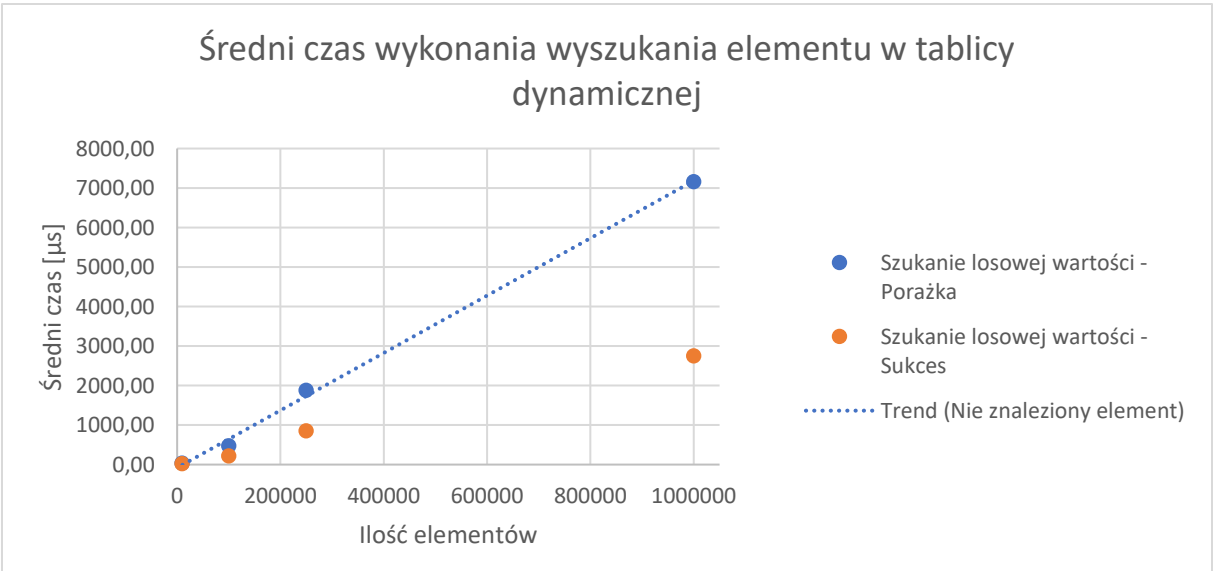
Wielkość listy			
10000	100000	250000	1000000
Porażka			
Średni czas (mikrosekundy)			
35,60	476,47	1880,80	7163,50

Wielkość listy			
10000	100000	250000	1000000
Sukces			
Średni czas (mikrosekundy)			
20,48	216,56	851,31	2751,31

Wykresy


Średni czas wykonania dodawania elementu w tablicy dynamicznej
Średni czas [µs]
5000,00
4500,00
4000,00
3500,00
3000,00
2500,00
2000,00
1500,00
1000,00
500,00
0,00
0
200000
400000
600000
800000
1000000
Ilość elementów
Dodawanie na przodzie listy
Dodawanie na tyle listy
Dodawanie w środku listy
Trend (koniec listy)
Trend (początek listy)
Trend (środek listy)


Średni czas wykonania usuwania elementu w tablicy dynamicznej
Średni czas [µs]
5000,00
4500,00
4000,00
3500,00
3000,00
2500,00
2000,00
1500,00
1000,00
500,00
0,00
0
200000
400000
600000
800000
1000000
Ilość elementów
Usuwanie na przodzie listy
Usuwanie na tyle listy
Usuwanie w środku listy
Trend (koniec listy)
Trend (początek listy)
Trend (środek listy)


Średni czas wykonania wyszukania elementu w tablicy dynamicznej
Średni czas [µs]
8000,00
7000,00
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
0
200000
400000
600000
800000
1000000
Ilość elementów
Szukanie losowej wartości - Porażka
Szukanie losowej wartości - Sukces
Trend (Nie znaleziony element)

Wnioski

Lista jest strukturą danych, której zaletą jest złożoność obliczeniowa stała dodawania i usuwania elementów na swoim końcu oraz początku. Lista dwukierunkowa ma zaletę prostej nawigacji w obie strony listy w przeciwieństwie do listy jednokierunkowej, ale koszt pamięci potrzebny do każdego elementu jest zwiększony przez konieczność dodania zmiennej wskazującej na adres poprzedniej wartości. W przeciwieństwie do tablicy, lista nie ma szybkiego dostępu do elementu umieszczonym w konkretnym miejscu, ale szybki dostęp do modyfikowania elementów na przodzie i tyle.

4. Kopiec typu maksimum

Opis

Kopiec binarny jest strukturą danych reprezentującą drzewo binarne. Podstawowym elementem kopca jest węzeł. Wszystkie węzły kopca są powiązane z innymi elementami kopca : zawierają element rodzica oraz 2 elementy będące potomkami danego węzła : Prawym i Lewym. Wyjątkiem są liście kopca – wartości nie posiadające w tym momencie żadnych dzieci, oraz korzeń kopca nie posiadający rodzica.

Kopiec binarny jest wyrażony tablicą, jednak elementy są powiązane w inny sposób. Element na indeksie 0 jest korzeniem kopca. Elementy na indeksach 1 oraz 2 są kolejno lewym i prawym dzieckiem kopca. Dla elementu i jej dziećmi będą elementy na indeksie $2i + 1$ oraz $2i + 2$, a jej rodzic będzie elementem o indeksie $\frac{i}{2} - 1$. Założeniem kopca binarnego typu maksimum jest: Każdy rodzic musi zawierać większą, bądź równą wartość niż jego dzieci. Oznacza to że w korzeniu zawsze będzie wartość maksymalna

Do podstawowych operacji na kopcu należą:

- Dodawanie elementy
  - Na końcu listy, którą wyrażony jest kopiec
    - Złożoność obliczeniowa: $\Theta(\log n)$
- Usuwanie elementu
  - Korzenia
    - Złożoność obliczeniowa: $\Theta(\log n)$
  - Najmłodszego elementu kopca
    - Złożoność obliczeniowa: $\Theta(1)$
  - W środku kopca
    - Złożoność obliczeniowa: $\Theta(\log n)$
- Wyszukanie elementu
  - złożoność obliczeniowa : Θ(i), gdzie i to indeks elementu wyszukiwanego, rzeczywista złożoność obliczeniowa : Θ(n)
- Operacja Build-Heap (naprawianie kopca)
  - Złożoność obliczeniowa: $\Theta(\mathrm{n})$
- Operacja Heapify (naprawiająca kopiec, biorąca pod uwagę wyłącznie element o indeksie podanym w argumencie)
  - Złożoność obliczeniowa: $\Theta(\log n)$

Opis operacji

W przypadku kopca, który wyrażony jest tablicą, nie jest potrzebna oddzielna klasa jak w przypadku drzew binarnych. Wszystkie operacje są wykonywane na wartościach zapisanych w tablicy, więc zamiast deklaracji nowego elementu, realokowana jest pamięć przeznaczona na tą tablicę. Element ten następnie jest zamieniany z właściwymi elementami, aż nie zostanie umieszczony w odpowiednim miejscu.

Element dodawany jest zawsze na końcu tablicy. Po jego umieszczeniu jego wartość jest porównywana z wartością rodzica i element jest zamieniany, jeśli wartość ta jest większa. Ta akcja powtarza się, aż wartość rodzica dodawanego elementu nie będzie większa niż wartość elementu dodawanego do kopca. Wstawienie elementu mniejszego od innych kończy się natychmiastowo, a większego niż wszystkie inne, przenosi element przez wszystkie poziomy kopca.

W przypadku usuwania elementu, jest on zamieniany z wartością końcową, po czym wykonywana jest operacja Heapify dla indeksu argumentu usuwanego (0 w przypadku korzenia), po czym pamięć przeznaczona na kopiec zostanie zrealokowana. Oznacza to że tak długo jak wartość przynajmniej jedno z dzieci elementu poprawianego jest większa niż wartość elementu poprzednio ostatniego, element ten zostanie zamieniony z dzieckiem o większej wartości i operacja Heapify zostanie wywołana ponownie, dla nowego indeksu zamienionego elementu. Jeśli element będzie liściem (nie będzie posiadał żadnych dzieci) operacja przestanie się wykonywać. Jeśli element jest ostatnim elementem kopca, sytuacja jest identyczna jak w tablicy.

Wyszukanie elementu w kopcu wykonywane przez porównywanie wartości kolejnych elementów w tablicy, przez którą wyrażony jest kopiec. Zachodzi identycznie jak w przypadku tablicy. Jednak przy wyszukiwaniu elementu o maksymalnej wartości, operacja wykonana jest natychmiastowo – z założeń kopca maksymalnego wynika że ten element będzie korzeniem kopca.

Operacja Build-Heap polega na wykonaniu operacji Heapify na elementach kopca, które nie są liściami, zaczynając od ostatniej wartości, która posiada potomka. Ostatni element posiadający potomka ma indeks $\frac{n}{2} - 1$. Operacja ta, wykonana na nieuporządkowanej tablicy, ustawi elementy w odpowiedniej kolejności, aby założenia kopca były zachowane.

Operacja Heapify, porównuje wartości elementu o indeksie podanym w argumencie i wartości jego dzieci, po czym zamienia jego miejsce w tablicy z miejscem większego z jego dzieci, o ile ta wartość jest większa niż wartość elementu o indeksie podanym. Jeśli nie nastąpiła zamiana albo element nie ma dzieci operacja kończy się, natomiast jeśli wystąpiła, operacja wykonuje samą siebie. Argumentem tej rekurencji jest indeks dziecka z którym nastąpiła zamiana elementu.

Wyniki

Poniżej znajdują się tabele przedstawiające zmierzone czasy wykonywania operacji na kopcu, oraz tabela przedstawiającą średnie czasów wykonania operacji.

Wstawianie											
Element większy niż wszystkie inne elementy				Element mniejszy niż wszystkie inne elementy				Losowy element			
Wielkość kopca				Wielkość kopca				Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych											
1 - 100000		1 - 1000000		1 - 100000		1 - 1000000		1 - 100000		1 - 1000000	
Czas w mikrosekundach				Czas w mikrosekundach				Czas w mikrosekundach			
0,4	0,3	0,5	0,7	0,2	0,2	0,8	0,5	0,3	0,3	0,9	0,5
0,3	0,5	0,8	0,6	0,2	0,2	0,2	0,5	0,2	0,3	0,6	0,5
0,3	0,3	1,1	0,6	0,2	0,3	0,4	0,8	0,2	0,7	0,6	0,4
0,2	0,4	0,5	0,6	0,1	0,3	0,6	0,5	0,2	0,1	0,7	0,5
0,2	0,5	0,3	0,6	0,1	0,1	0,1	0,5	0,2	0,2	0,8	0,4
0,3	0,3	0,4	0,5	0,2	0,2	0,3	0,4	0,1	0,2	0,6	0,7
0,3	0,3	0,4	0,7	0,2	0,5	0,3	0,5	0,1	0,6	0,6	0,7
0,3	0,6	0,5	0,4	0,1	0,3	0,1	0,5	0,2	0,4	0,8	0,4
0,2	0,3	1	1,3	0,2	0,2	0,2	0,4	0,1	0,4	0,3	0,6
0,5	0,4	0,9	0,5	0,1	0,3	0,2	0,4	0,1	0,3	0,6	0,7
0,5	0,3	0,4	0,4	0,1	0,2	0,3	0,4	0,1	0,4	0,9	1,4
0,4	0,3	0,6	0,5	0,1	0,3	0,3	0,3	0,2	0,5	0,2	1
0,3	0,5	0,4	0,5	0,2	0,3	1,8	0,5	0,1	0,2	0,8	0,5
0,3	0,3	0,7	0,6	0,1	0,1	0,6	0,6	0,1	0,3	1,2	1,1
0,2	0,3	3,5	0,9	0,1	0,2	0,2	0,8	0,1	0,6	0,7	0,6
0,2	0,4	0,4	1,2	0,1	0,1	0,3	1,1	0,2	0,1	0,5	0,4
0,3	0,4	0,5	0,5	0,2	0,2	0,7	0,4	0,1	0,2	0,8	0,8
0,3	0,3	0,4	1,1	0,2	0,4	0,2	0,9	0,2	0,2	0,6	0,5
0,3	0,3	1,7	0,8	0,2	0,3	0,2	0,7	0,1	0,3	0,3	0,4
0,2	0,5	0,5	0,5	0,2	0,1	0,2	0,7	0,1	0,3	0,6	0,4

Usuwanie							
Korzeń				Koniec			
Wielkość kopca				Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych							
1 - 100000		1 - 1000000		1 - 100000		1 - 1000000	
Czas w mikrosekundach				Czas w mikrosekundach			
0,3	0,8	0,6	1,1	0,1	0,1	0,2	0,6
0,3	0,3	0,6	0,7	0,1	0,1	0,4	1
0,3	0,4	0,5	1,1	0,1	0,2	0,3	0,8
0,3	0,3	0,7	0,9	0,1	0,2	0,4	0,8
0,2	0,4	1,1	0,8	0,1	0,1	0,1	0,9
0,2	0,3	0,5	0,6	0	0,1	0,3	0,7
0,3	0,3	0,8	0,8	0,1	0,5	0,3	0,4
0,2	0,3	0,5	1,1	0,1	0,1	0,1	0,3
0,2	0,4	0,5	1,3	0	0,1	0,2	0,3
0,2	0,4	0,5	0,7	0,1	0,2	0,3	0,7
0,2	0,5	0,5	0,8	0,1	0,4	0,4	0,5
0,2	0,4	0,5	0,7	0,1	0,1	0,2	0,4
0,2	0,4	0,4	0,8	0	0,3	0,3	0,3
0,2	0,8	0,5	0,9	0,1	0,2	0,4	0,5
0,2	0,4	0,5	0,8	0	0,1	0,9	0,3
0,3	0,3	0,4	1,1	0,1	0,2	0,4	0,4
0,2	0,3	0,4	0,7	0,1	0,2	0,1	0,3
0,3	1	0,6	1,5	0,1	0,1	0,3	0,4
0,3	0,5	0,7	0,8	0,1	0,3	1,3	0,5
0,2	0,3	0,6	0,7	0,1	0,1	0,2	0,3

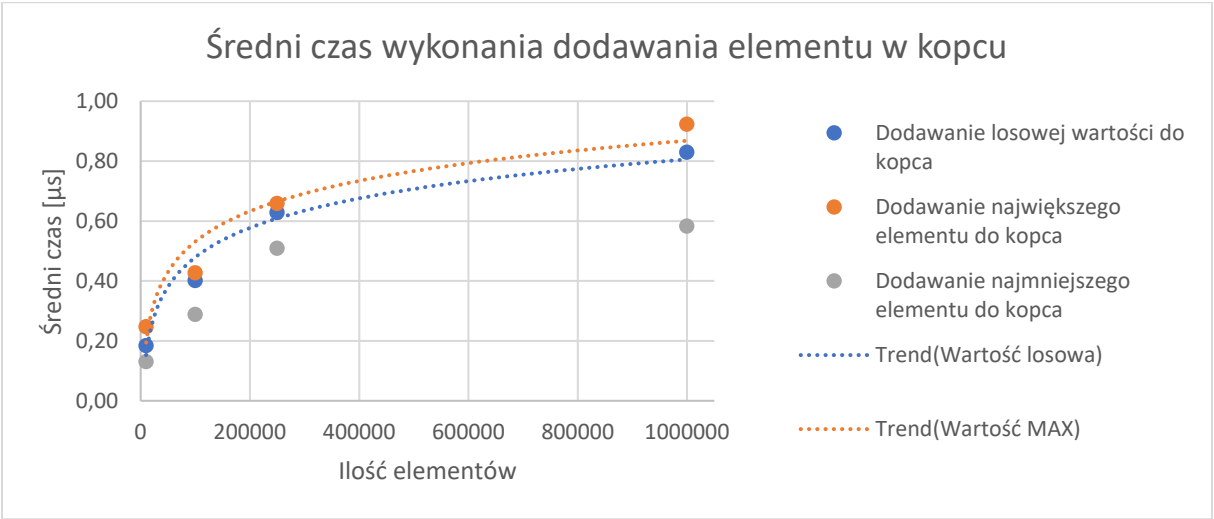
Szukanie							
Losowa wartość							
Wielkość kopca							
10000		100000		250000		1000000	
Zasięg liczb losowych							
1 - 100000				1 - 10000000			
Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?
10,3	nie	101,2	nie	253,5	nie	1135,3	nie
10,9	nie	107,6	nie	264,6	nie	1067,2	nie
10,6	nie	101,4	nie	258,6	nie	646,6	tak
10,4	nie	22,5	tak	135,1	tak	1014,7	nie
10,3	nie	101,2	nie	258,4	nie	498,6	tak
10,4	nie	106,4	nie	261,7	nie	628,6	tak
12,1	nie	101,2	nie	210,6	tak	601,2	tak
10,3	nie	103	nie	225,6	tak	554,8	tak
10,4	nie	108,1	nie	258,5	nie	326,5	tak
8,7	tak	105,8	nie	258,5	nie	1040	nie
10,3	nie	105,8	nie	311,5	nie	1058,3	nie
10,3	nie	103,5	nie	256,1	nie	824,8	tak
10,4	nie	101,1	nie	268	nie	1060,8	nie
10,4	nie	101,1	nie	252,7	nie	85,3	tak
10,3	nie	105,6	nie	136,7	tak	766,4	tak
10,3	nie	101,2	nie	259,2	nie	46,8	tak
10,3	nie	108,4	nie	252,9	nie	1248,1	nie
10,3	nie	101,1	nie	363,5	nie	1022,1	nie
11,9	nie	101,1	nie	211	tak	606,6	tak
10,3	nie	101,1	nie	252,9	nie	190,7	tak

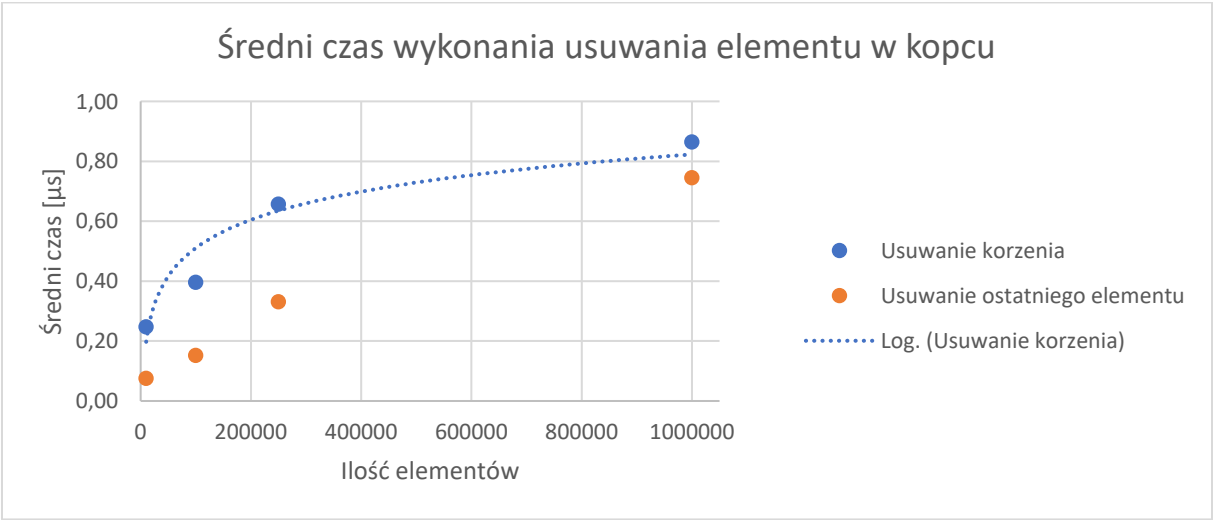
Szukanie				Naprawianie stosu (Build-Heap)			
Maksymalna wartość				---			
Wielkość kopca				Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych				Zasięg liczb losowych			
1 - 100000		1 - 1000000		1 - 100000		1 - 1000000	
Czas w mikrosekundach				Czas w mikrosekundach			
0,1	0,1	0,1	0,1	153,1	1563,1	4072,9	15980,3
0	0	0	0	149,6	1708,5	4243,2	16361,4
0	0	0	0	154,5	1522,1	4254,1	15909,2
0,1	0,1	0	0	151,2	1546,7	4146,1	16270,3
0,1	0	0,1	0	158,6	1724,7	4222,6	16487,2
0	0	0,1	0	154,6	1610,1	4101,6	17259,4
0,1	0	0	0	155,7	1585,6	4183,5	16177,8
0,1	0,1	0,1	0	167,9	1600,2	4222,4	16495,9
0	0	0	0,1	149,3	1610	4298,4	16094,1
0	0	0,1	0,1	152,2	1626,2	5037,6	16398,2
0	0	0,1	0	152,2	1602,2	4199,6	16172,2
0	0	0,1	0	150,4	1509,2	4232,5	17001
0	0	0	0,1	150,4	1566,8	4259,5	16828,1
0	0	0	0	150,4	1572,4	4301,5	16677,2
0,1	0	0	0	163,2	1593,9	4110,1	16342,1
0	0,1	0	0	153,7	1587,3	4323,6	16489,3
0	0	0	0	154,1	1583,6	4353,7	17704,7
0,1	0	0	0	158,9	1771,3	4078,9	17418,1
0	0	0	0	159,5	1522,5	4179,7	17615,8
0	0,1	0,1	0,1	160,2	1588,6	4193,6	17451,1

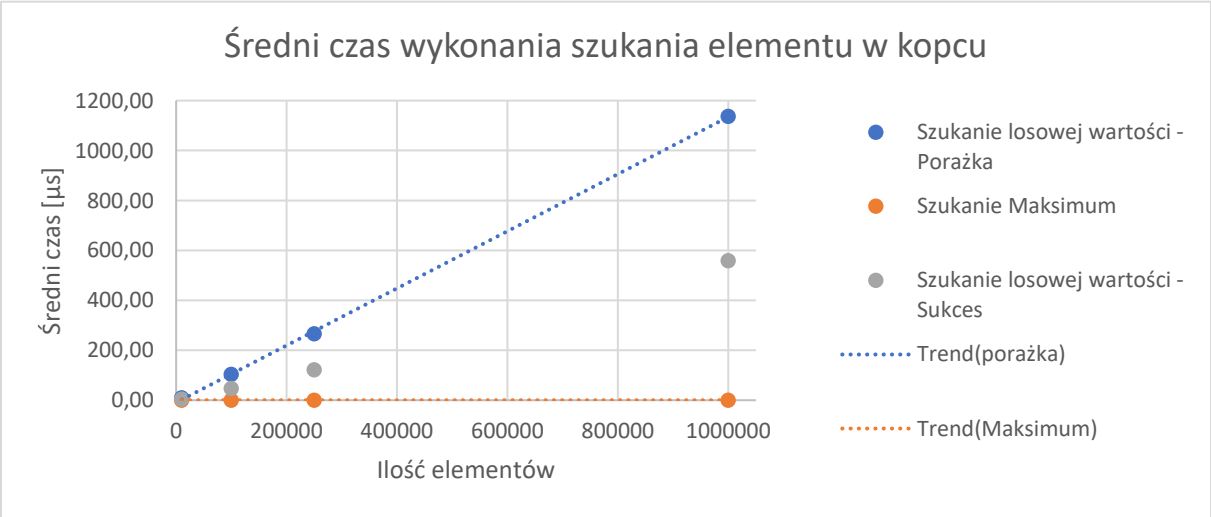
Średnie czasy pomiarów

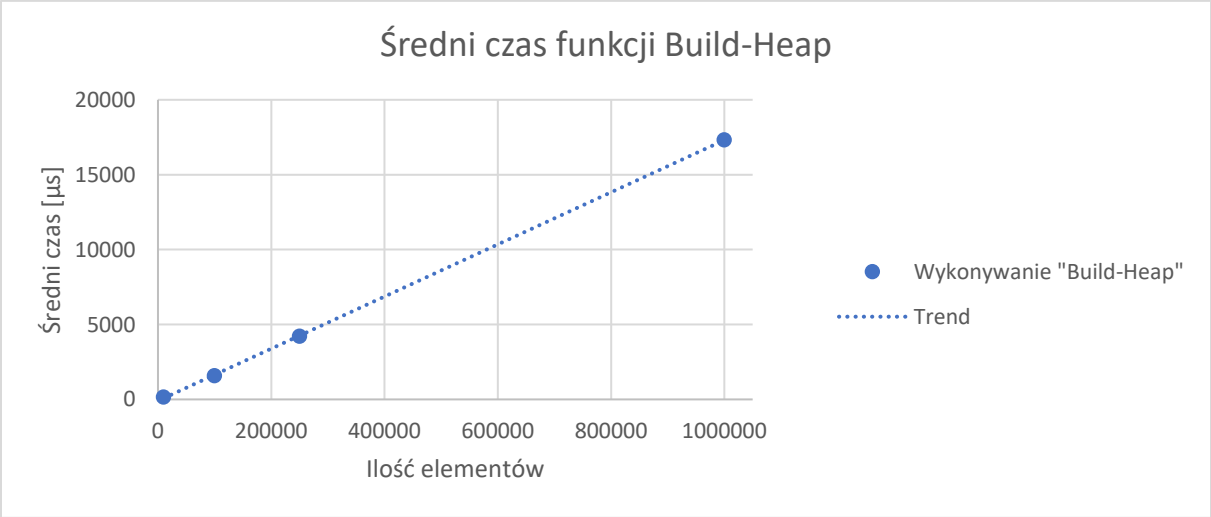
Wstawianie												
Losowy element				Element większy niż wszystkie inne elementy				Element mniejszy niż wszystkie inne elementy			
Wielkość kopca				Wielkość kopca				Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,19	0,40	0,63	0,83	0,25	0,43	0,66	0,92	0,13	0,29	0,51	0,58
Usuwanie								Naprawianie stosu (Build-Heap)			
Korzeń				Koniec				---			
Wielkość kopca				Wielkość kopca				Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,25	0,40	0,66	0,87	0,076	0,15	0,33	0,75	158,323	1585,977	4227,849	17320,299
Szukanie											
Losowa wartość								Maksymalna wartość			
Wielkość kopca								Wielkość kopca			
10000	100000	250000	1000000	10000	100000	250000	1000000	10000	100000	250000	1000000
Porażka				Sukces				Średni czas (mikrosekundy)			
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)				0,019	0,033	0,032	0,031
10,44	103,73	266,90	1136,96	4,60	47,89	122,21	558,70				

Wykresy









W przypadku usuwania ostatniego elementu kopca, czas poświęcony rośnie w sposób przypominający wzrost logarytmiczny, zamiast stały. Może to być spowodowane czasem, który operacja poświęca na realokacje pamięci przeznaczonej na kopiec, dla tak wielkiej ilości elementów.

Wnioski

Kopiec typu maksimum jest strukturą danych pozwalającą na szybkie znalezienie elementu o maksymalnej wartości. Jako że czas usuwania i dodawania elementów przeciętnie będzie rósł w skali logarytmicznej, kopiec jest wystarczająco efektywny pod tym względem, jednak dodawanie nie oferuje takiej samej elastyczności jak w przypadku tablicy bądź listy. Jednak usuwanie środkowego elementu jest bardziej efektywne niż w przypadku tych dwóch struktur danych.

Ze względu na fakt, że w korzeniu kopca zawsze znajduje się element o maksymalnej wartości, kopiec jest używany do kolejki priorytetowej – zakładając że priorytetem elementu jest wartość, natychmiastowo można znaleźć element o kolejnym największym priorytecie i usunąć go z „kolejki” oraz naprawić kopiec używając operacji Heapify na indeksie korzenia.

5. Drzewo Czerwono-Czarne

Opis

Drzewo Czerwono-Czarne jest typem samo-balansującego się drzewa przeszukiwań binarnych, podobnym do drzew BST, ale zawierającym dodatkowe reguły co do balansowania drzewa. Te reguły upewniają się że drzewo zawsze będzie głębokości $\log_2 n$, gdzie n to liczba elementów drzewa. Ograniczona głębokość drzewa, zapewnia lepszą wydajność przy operacjach takich jak dodawanie, usuwanie czy wyszukiwanie elementu.

Podstawowym elementem drzewa jest węzeł, zawierający wartość, informację o swoim kolorze, oraz wskaźniki na swojego rodzica i swoje dzieci. Każdy węzeł może być czerwony albo czarny, może mieć swoje dzieci i musi mieć rodzica (chyba że jest korzeniem drzewa). W mojej implementacji taki element przedstawia klasa NodeRB, zawierająca swoje własności i funkcje.

Aby zaoszczędzić pamięć, wszystkim pustym wskaźnikom dzieci (tzn. takim, dla których elementy jeszcze nie istnieją) zostaje przypisany wskaźnik na jeden węzeł zwany nullNode, oznaczający że ta wartość na razie jest pusta. nullNode jest koloru czarnego, aby w przypadku modyfikacji drzewa, ułatwić dobranie przypadku, jaki trzeba wykonać aby zachować własności drzewa.

Każdy nowy węzeł jest koloru czerwonego.

Własnościami drzewa czerwono-czarnego są:

- Każdy węzeł jest czerwony albo czarny
- Korzeń jest koloru czarnego
- Każdy liść (węzeł bez potomków) jest czarny – nullNode traktowany jest jako liść
- Jeśli węzeł jest koloru czerwonego, jego potomkowie muszą być czarni
- Dla każdego węzła wszystkie proste ścieżki od niego do liści zawierają taką samą liczbę czarnych węzłów.

Te warunki zapewniają, że drzewo czerwono-czarne będzie zrównoważone. Jeśli przynajmniej jedno z tych założeń zostanie złamane, drzewo czerwono-czarne wykona operację przywracającą ich zgodność.

Do podstawowych operacji na drzewie czerwono-czarnym należą:

- Dodawanie elementu
  - Złożoność obliczeniowa: $\Theta(\log n)$
- Usuwanie elementu
  - Złożoność obliczeniowa: $\Theta(\log n)$
- Szukanie elementu
  - Złożoność obliczeniowa: $\Theta(\log n)$
- Rotacja prawa, Rotacja lewa
  - Złożoność obliczeniowa: $\Theta(1)$

Opis operacji

W drzewie czerwono-czarnym dodawanie i usuwanie elementów jest o wiele bardziej skomplikowane niż w poprzednich strukturach danych, jest o wiele więcej przypadków które musimy wziąć pod uwagę. Miejsce elementu określone jest przez jego wartość: zaczynając od korzenia porównujemy element dodawany z aktualnie branym pod uwagę węzłem, jeśli element dodawany ma mniejszą wartość, porównujemy dodawany element z lewym dzieckiem aktualnie branego pod uwagę elementu; jeśli ma większą wartość porównujemy go z prawym dzieckiem; jeśli to dziecko jest nullNode'em, wtedy jego miejsce w drzewie zajmuje dodawany węzeł.

Jeśli po tej operacji nie są zachowane założenia drzewa czerwono-czarnego program wykonuje operację w zależności od powstałej sytuacji. Jeśli element jest pierwszym elementem drzewa, oznacza to że jest korzeniem – jego kolor jest zmieniany na czarny. Jeśli rodzic elementu jest koloru czarnego, własności drzewa nie zostają naruszone, operacja dodawania kończy się. Jeśli rodzic elementu jest koloru czerwonego, wykonywana jest jedna z następujących operacji (operacje opisane są dla wypadku, gdy rodzic dodawanego elementu jest prawym dzieckiem swojego rodzica; W wypadku gdy jest on lewym dzieckiem operacje wykonywane są odbiciem lustrzanym operacji wypadku gdy jest prawym):

1. Jeśli stryj elementu dodawanego jest koloru czerwonego, rodzic elementu, stryj oraz rodzic ojca zmieniają kolory: rodzic i stryj – na czarny, a prarodzic – na czerwony. W następnej pętli operacji będziemy sprawdzać czy prarodzic nie łamie własności drzewa
2. Jeśli stryj elementu dodawanego jest koloru czarnego, a element dodawany jest prawym dzieckiem swojego ojca, przypisuje się ojcowi kolor czarny, praojcowi – czerwony i wykonuje się lewą rotację praojca elementu
3. Jeśli stryj elementu dodawanego jest koloru czarnego, a element dodawany jest lewym dzieckiem swojego ojca, elementem sprawdzanym staje się rodzic elementu dodawanego i wykonuje się na nim rotację w prawo, aby sprowadzić sytuację do sytuacji 2.

Operacje te są wykonywane w pętli, tak długo jak rodzic elementu sprawdzanego jest koloru czerwonego. Na końcu każdej iteracji, kolor korzenia zostaje zmieniony na czarny, aby uniknąć naruszenia właściwości drzewa. Jak wspomniałem powyżej – jeśli rodzic elementu sprawdzanego jest lewym dzieckiem swojego rodzica, operacje wykonywane są niemal identyczne, tylko są odbiciem lustrzanym operacji opisanych.

W przypadku usuwania elementu mogą występować następujące sytuacje:

1. Jeżeli węzeł nie ma dzieci i jest koloru czerwonego to można go usunąć bez naruszania założeń drzewa RB.
2. Jeżeli węzeł bez potomków jest koloru czarnego, założenia zostaną naruszone.
3. Jeżeli ma tylko 1 potomka zamienia z nim element usuwany
4. Jeżeli ma 2 potomków zamienia węzeł usuwany z poprzednikiem, bądź następnikiem.

Następnie należy wykonać operacje przywracające założenia drzewa (tzn. kiedy kolor usuwanego elementu był czarny), podobnie jak przy dodawaniu operacja zależy od tego czy element usuwany był prawym, czy lewym potomkiem (operacje ponownie są prawie identyczne – są swoim odbiciem). Podane operacje są opisane w wypadku jeśli węzeł usuwany był prawym dzieckiem:

1. Brat elementu jest koloru czerwonego – zamieniamy kolory brata i rodzica i wykonujemy prawą rotację na rodzicu. Sprowadza to przypadek do jednego z kolejnych przypadków.
2. Brat elementu jest koloru czarnego i potomstwo brata jest koloru czarnego – ustawiamy kolor brata na czerwony, i zmieniamy element obserwowany na rodzica poprzedniego elementu.
3. Brat elementu jest koloru czarnego, lewe dziecko brata jest czarne a prawe czerwone – zmieniamy kolory potomstwa brata oraz wykonujemy rotację lewą na bracie elementu. Sprowadza to przypadek do przypadku 4
4. Brat elementu jest koloru czarnego, lewe dziecko brata jest czerwone, a prawe czarne – zamieniamy kolor lewego dziecka na czarny, rodzica elementu „obserwowanego” na czarny, wykonujemy prawą rotację na tym rodzicu i możemy skończyć pętle naprawiającą drzewo

Operacje te wykonują się w pętli, która trwa tak długo jak elementem obserwowanym nie jest korzeń, albo kolorem elementu obserwowanego nie jest czarny. Element obserwowany to element odnośnie którego w danej iteracji wykonujemy operację. Początkowym elementem obserwowanym jest lewe bądź prawe dziecko elementu usuwanego – te które zastąpiło element usuwany.

Szukanie zachodzi identycznie jak w drzewie BST – porównując odpowiednie dzieci (lewe jeśli wartość elementu poszukiwanego jest mniejsza od sprawdzanego; prawe jeśli jest większa) przechodzimy przez poziomy drzewa aż nie napotkamy poszukiwanego elementu albo Nil’a – co kończy operację poszukiwania.

Wyniki

Poniżej znajdują się tabele przedstawiające zmierzone czasy wykonywania operacji na drzewie, oraz tabela przedstawiającą średnie czasów wykonania operacji. Liczby losowe przy usuwaniu były w odpowiednim zasięgu, aby prawie pewnym było że istnieje element, który chcemy usunąć.

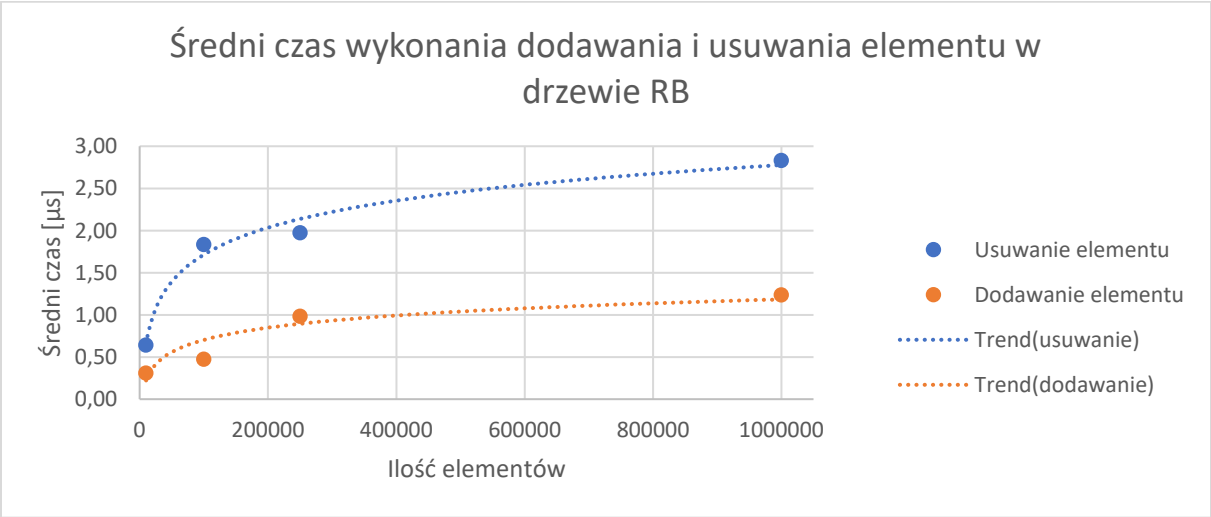
Wstawianie				Usuwanie			
Losowa wartość				Losowa wartość			
Wielkość drzewa				Wielkość drzewa			
10000	100000	250000	1000000	10000	100000	250000	1000000
Zasięg liczb losowych							
1 - 100000		1 - 1000000		1 - 1000	1 - 10000	1 - 25000	1 - 100000
Czas w mikrosekundach				Czas w mikrosekundach			
0,5	0,6	0,6	1	1,1	2	2,2	2,5
0,2	0,3	0,4	0,9	0,7	1,6	1,7	3
0,2	0,5	0,6	0,9	0,7	1,9	1,8	2,2
0,8	0,5	0,8	0,6	0,3	2,2	2,1	2,2
0,4	0,5	0,5	0,9	0,5	1,5	2,1	2
0,2	0,3	0,5	0,8	0,6	2,1	2,4	1,9
0,3	0,5	1,1	0,9	0,6	1,7	2	2,3
0,3	0,4	0,7	1,1	0,5	1,7	2,3	2,7
0,4	0,3	0,8	2,8	0,7	2,1	2,3	2,2
0,3	0,5	1,2	1,6	0,5	2	2,1	18,7
0,2	0,3	0,8	1,6	0,5	1,9	1,7	3,1
0,2	0,4	0,6	1,7	0,5	1,5	2,1	2,8
0,2	0,4	0,7	0,7	0,3	2,1	1,4	3,7
0,2	0,4	1,1	0,8	0,4	1,5	1,9	2,6
0,2	0,4	0,6	1	0,5	1,8	2	2,2
0,3	0,3	0,4	2,9	0,3	1,4	1,8	2,6
0,4	0,7	0,7	1	0,8	1,9	1,9	2,2
0,4	0,4	0,9	0,9	0,3	1,7	1,6	2,6
0,2	0,4	0,6	0,8	0,4	1,4	2,4	2,3
0,2	0,5	0,4	0,7	0,5	1,7	2	2,3

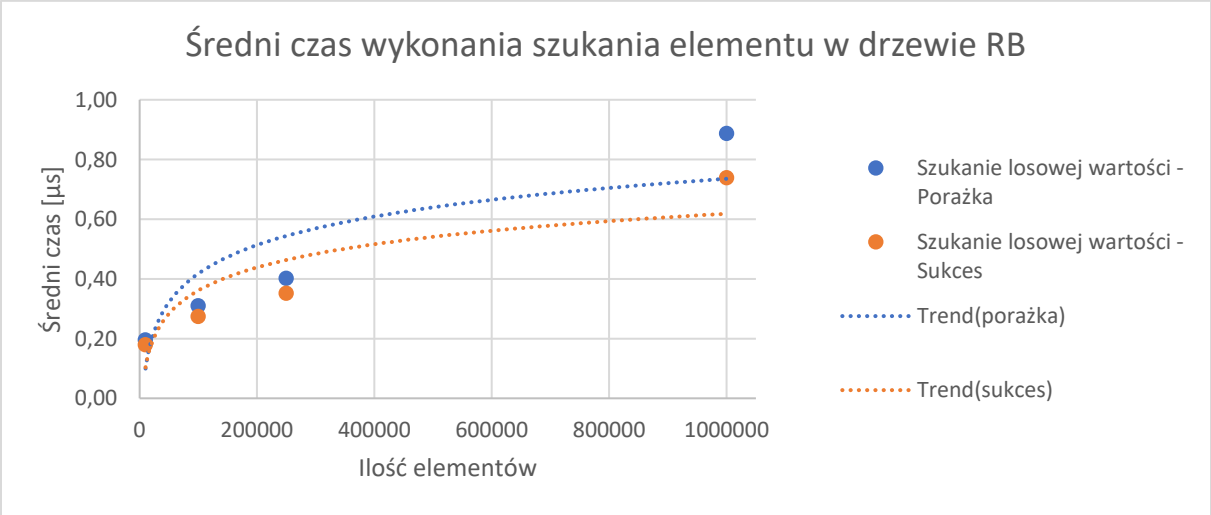
Szukanie							
Losowa wartość							
Wielkość drzewa							
10000		100000		250000		1000000	
Zasięg liczb losowych							
1 - 100000				1 - 1000000			
Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?	Czas	Czy znaleziono?
0,3	nie	0,8	tak	0,5	nie	0,7	tak
0,3	nie	0,2	tak	0,4	nie	0,7	nie
0,2	nie	0,2	tak	0,5	nie	0,7	tak
0,1	nie	0,2	tak	0,4	nie	0,5	tak
0,2	tak	0,3	nie	0,3	tak	0,7	tak
0,2	nie	0,3	tak	0,5	nie	0,8	nie
0,2	tak	0,3	tak	0,4	nie	1,2	nie
0,2	nie	0,2	tak	0,3	nie	0,6	tak
0,1	nie	0,3	nie	0,3	nie	0,5	tak
0,2	nie	0,3	tak	0,3	nie	0,9	tak
0,2	nie	0,2	tak	0,5	nie	1,1	nie
0,2	nie	0,1	tak	0,3	nie	0,7	nie
0,2	nie	0,3	nie	0,3	nie	0,8	nie
0,1	nie	0,2	tak	0,5	nie	0,9	tak
0,2	nie	0,4	nie	0,3	tak	0,9	nie
0,1	nie	0,2	tak	0,3	nie	0,7	tak
0,1	nie	0,4	tak	0,3	nie	0,5	tak
0,1	nie	0,2	tak	0,4	nie	0,7	nie
0,2	nie	0,2	tak	0,6	nie	0,5	tak
0,2	nie	0,3	tak	0,4	nie	0,9	tak

Średnie pomiary czasów

Wstawianie				Usuwanie			
Losowa wartość				Losowa wartość			
Wielkość drzewa				Wielkość drzewa			
10000	100000	250000	1000000	10000	100000	250000	1000000
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,31	0,48	0,98	1,24	0,64	1,84	1,97	2,83
Szukanie							
Losowa wartość							
Wielkość kopca							
10000	100000	250000	1000000	10000	100000	250000	1000000
Porażka				Sukces			
Średni czas (mikrosekundy)				Średni czas (mikrosekundy)			
0,20	0,31	0,40	0,89	0,18	0,27	0,35	0,74

Wykresy





Zaznaczone zostały trendy dla wszystkich operacji, gdyż czas wzrasta w przewidywany sposób (logarytmiczny). Wykroczenie w przypadku szukania elementu poza przewidywany wzrost jest prawdopodobnie spowodowane niepewnością sposobu pomiaru. Dłuższy czas usuwania jest spowodowany większą ilością operacji które funkcja musi wykonać

Wnioski

Drzewo czerwono czarne jest strukturą danych pozwalającą na szybkie szukanie elementów o podanym kluczu, co jest jej głównym zastosowaniem. Dodawanie oraz usuwanie elementów jest szybsze niż w przypadku dodawania elementu na początku, bądź w środku tablicy, w środku list i zajmuje podobną ilość czasu w kopcu. Jednak w przypadku gdy nie potrzebujemy wyszukiwać szczególnych elementów, tablica oraz lista mogą okazać się szybszymi strukturami, w przypadku dodawania dużej ilości elementów (np. na koniec tablicy).