STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA

Temat Projektu: **Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych**

Michał Salamon 259167

1. Wprowadzenie:

Mój projekt zawiera implementacje oraz pomiary czasu działania poszczególnych operacji w różnych strukturach danych, takich jak:

- Tablica Dynamiczna
- Lista Dwukierunkowa
- Kopiec Binarny
- Drzewo Czerwono-Czarne

Dla tablicy oraz listy operacje dodawania posiadają dodatkowe opcje. Można wybrać, z którego miejsca usunie lub na które miejsce wstawi się element. Kopiec oraz drzewo nie posiadają takich opcji. Dla kopca usuwanie odbywa się z końca struktury, a następnie wstawiany już przez sam program w odpowiednie miejsce. Drzewo natomiast element wstawia w odpowiednie dla niego miejsce według zasad obowiązujących w drzewie RB.

W menu każdej struktury jest możliwość zbadania czasu trwania każdej operacji oraz wyświetlenie go w 3 różnych jednostkach czasu (sekundy, milisekundy, mikrosekundy). Po każdorazowym zbadaniu czasu wyświetlane są wyniki.

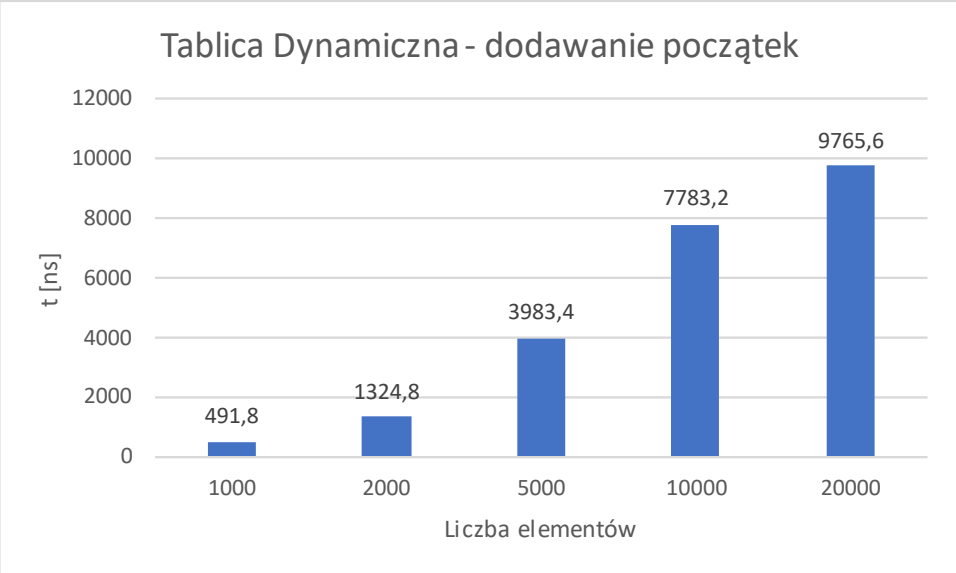
2. Poszczególne struktury:

Tablica dynamiczna:

Tablica była alokowana dynamicznie, tzn. zawsze zajmowała dokładnie tyle pamięci, ile danych przechowywała. Z racji na konieczność relokowania tablicy przy każdym dodaniu i usunięciu elementu średnia oraz pesymistyczna złożoność obliczeniowa tych operacji wynosiła O(n). Taką samą złożoność posiada operacja wyszukania elementu w strukturze.

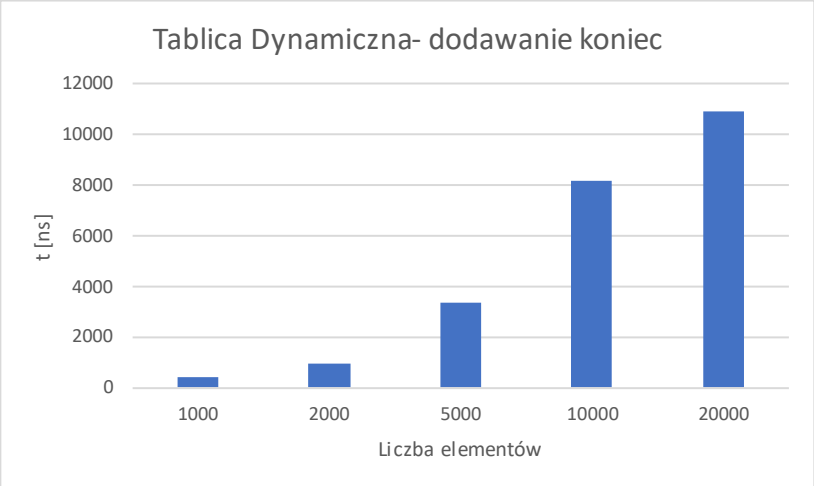
Dodawanie:

- Początek:



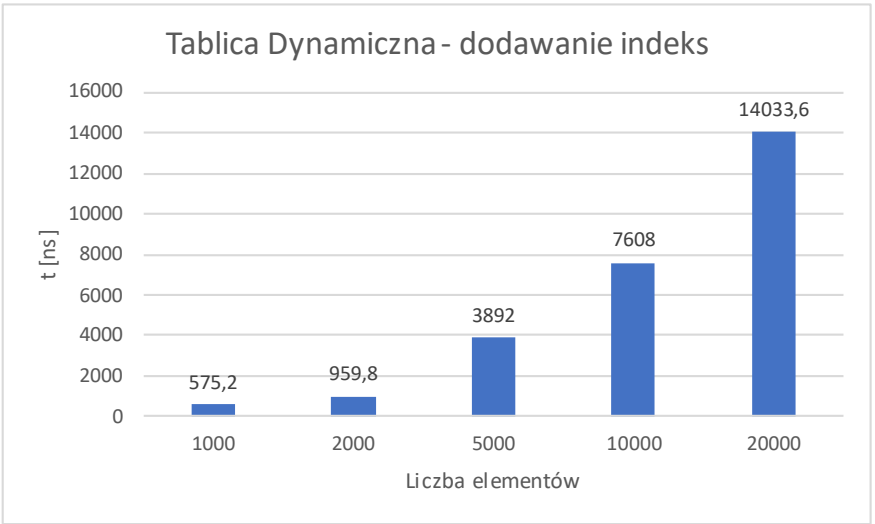
	Tablica - dodawanie początek				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	458	708	3583	6834	11206
	459	792	4250	7583	8834
	584	3375	3125	8291	9038
	458	708	6000	6708	7625
	500	1041	2959	9500	12125
Średni czas	491,8	1324,8	3983,4	7783,2	9765,6

- Koniec:



	Tablica - dodawanie koniec				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	458	584	4417	9125	7292
	500	542	3542	6833	17667
	375	334	2250	7750	8667
	375	2041	3542	10042	9709
	375	1250	2958	7166	11250
Średni czas	416,6	950,2	3341,8	8183,2	10917

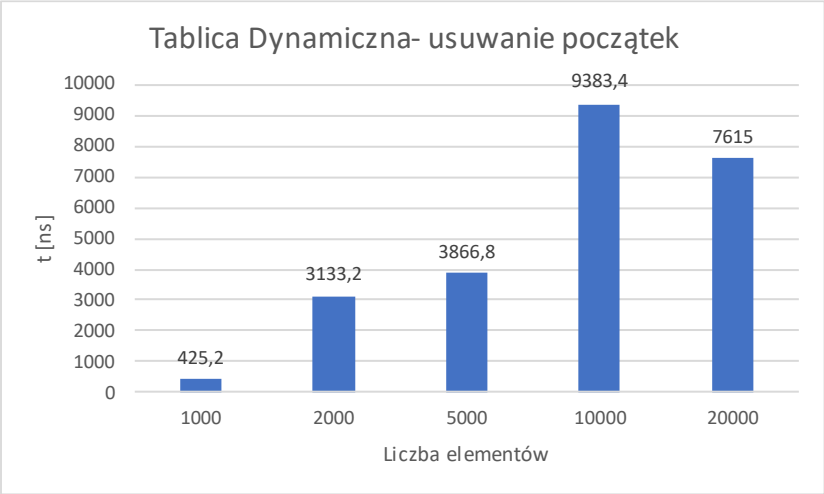
- Wybrane miejsce:



	Tablica - dodawanie indeks				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	542	958	3500	7125	6834
	667	1083	4251	8642	7375
	500	916	3084	8149	9834
	792	750	3417	6708	9375
	375	1092	5208	7416	36750
Średni czas	575,2	959,8	3892	7608	14033,6

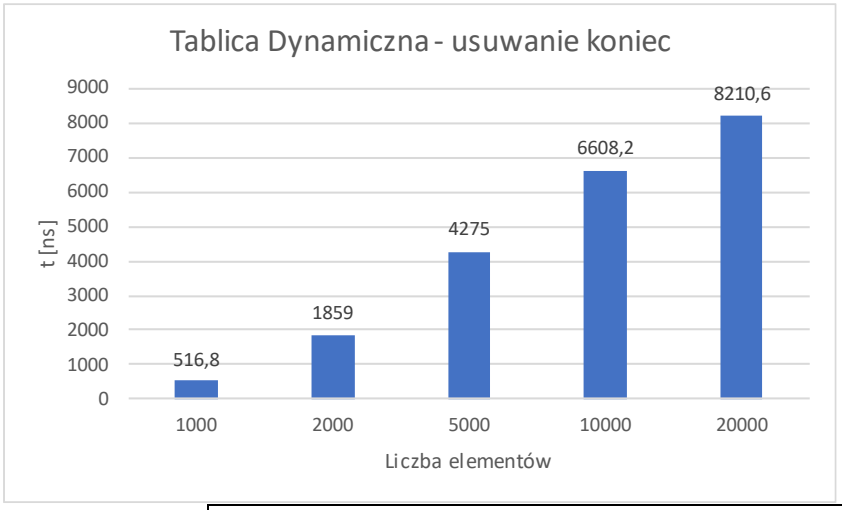
Usuwanie:

- Początek:



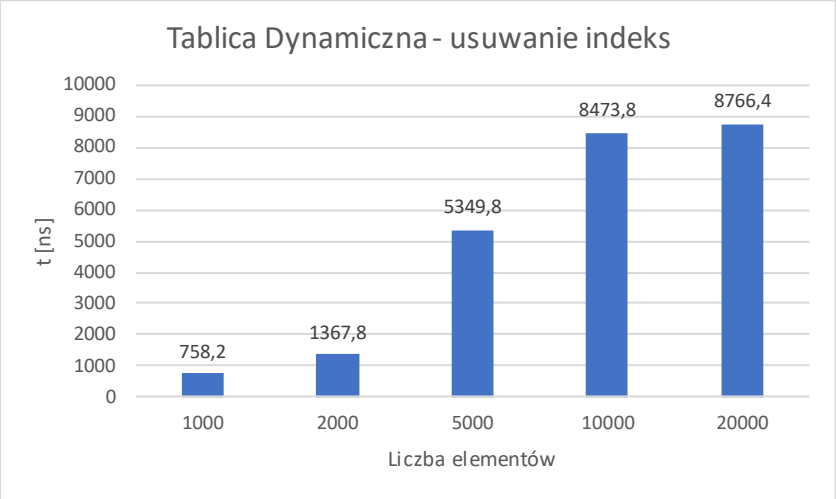
	Tablica - usuwanie początek				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	375	2708	3208	10333	6782
	417	3167	3208	7959	6917
	250	3333	4709	8791	8834
	500	3083	4167	10125	8917
	584	3375	4042	9709	6625
Średni czas	425,2	3133,2	3866,8	9383,4	7615

- Koniec:



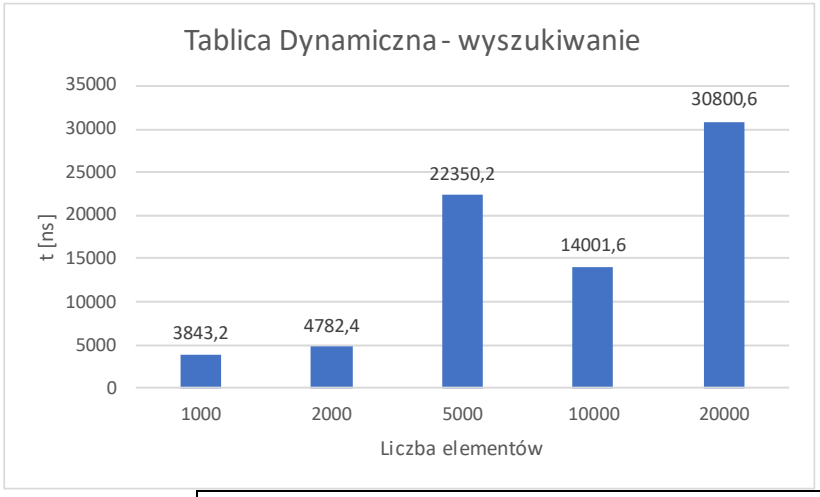
	Tablica - usuwanie koniec				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	459	2463	5084	7041	9929
	583	750	3250	7166	7083
	333	833	3708	6167	7500
	500	1833	6041	7292	7583
	709	3416	3292	5375	8958
Średni czas	516,8	1859	4275	6608,2	8210,6

- Wybrane miejsce:



	Tablica - usuwanie indeks				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	781	798	4875	9208	7291
	625	1208	5708	9280	8583
	969	1917	6541	7583	7167
	750	1833	4834	8298	11250
	666	1083	4791	8000	9541
Średni czas	758,2	1367,8	5349,8	8473,8	8766,4

Wyszukiwanie:



	Tablica - wyszukiwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	4508	8750	13834	35333	6166
	6708	4708	32709	7334	36712
	2208	2000	32083	6792	22084
	3250	5912	2000	6666	47291
	2542	2542	31125	13883	41750
Średni czas	3843,2	4782,4	22350,2	14001,6	30800,6

Tablica Dynamiczna, złożoność obliczeniowa (elementy ustawione po kolei w pamięci):

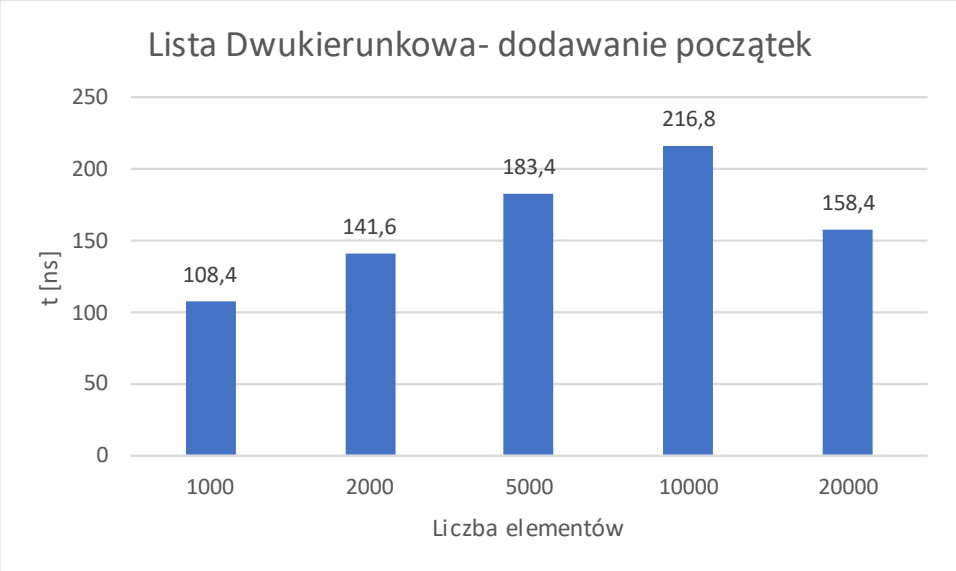
- Dodawanie elementów początek, koniec: **O(n)**
- Usuwanie elementów początek, koniec: **O(n)**
- Wyszukiwanie elementu: **O(n)**

Lista dwukierunkowa:

Lista to struktura służąca do reprezentacji zbiorów dynamicznych, w której elementy ułożone są w liniowym porządku. Każdy element listy składa się z co najmniej trzech pól: klucza, pola wskazującego na poprzedni element listy oraz pola wskazującego na następny element listy.

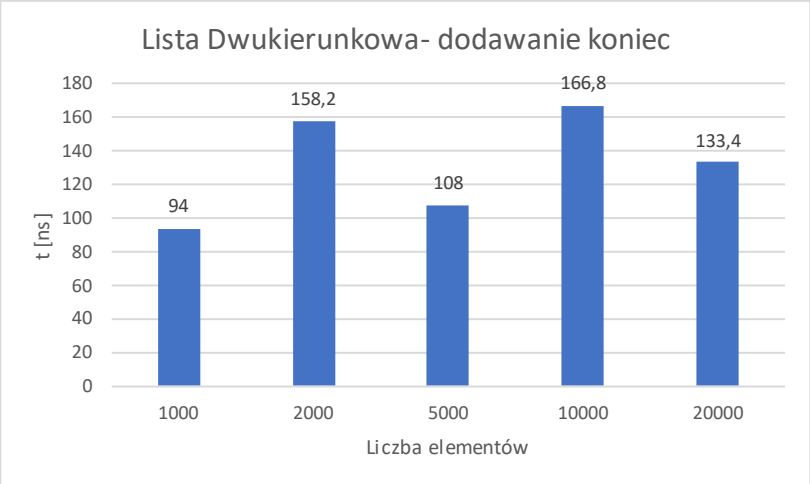
Pesymistyczny czas dodawania elementów na początek i koniec listy jest równy O(1), a w wybrane miejsce O(n). Tak samo jest z usuwaniem. Wyszukiwanie ma czas rzędu O(n).

- Początek:



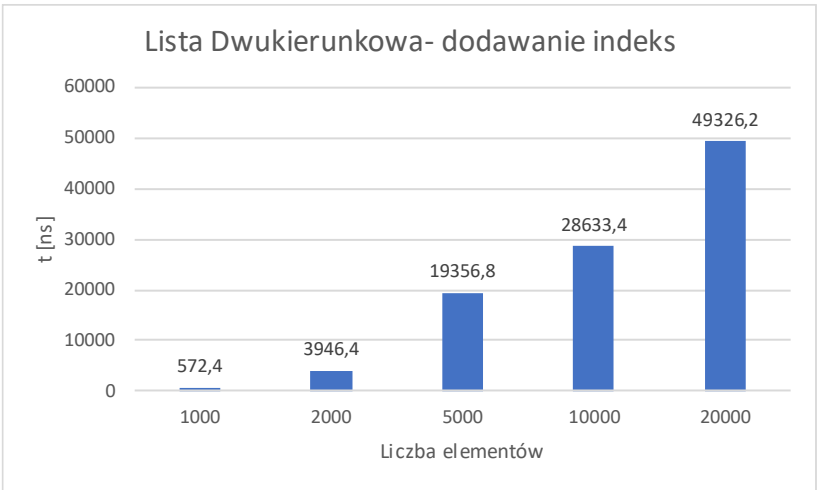
	Lista - dodawanie początek				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	83	125	500	125	84
	84	209	125	542	125
	125	166	83	125	125
	125	125	84	167	167
	125	83	125	125	291
Średni czas	108,4	141,6	183,4	216,8	158,4

- Koniec:



	Lista - dodawanie koniec				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	125	250	166	167	125
	125	167	83	167	125
	125	166	83	166	166
	83	125	125	167	125
	12	83	83	167	126
Średni czas	94	158,2	108	166,8	133,4

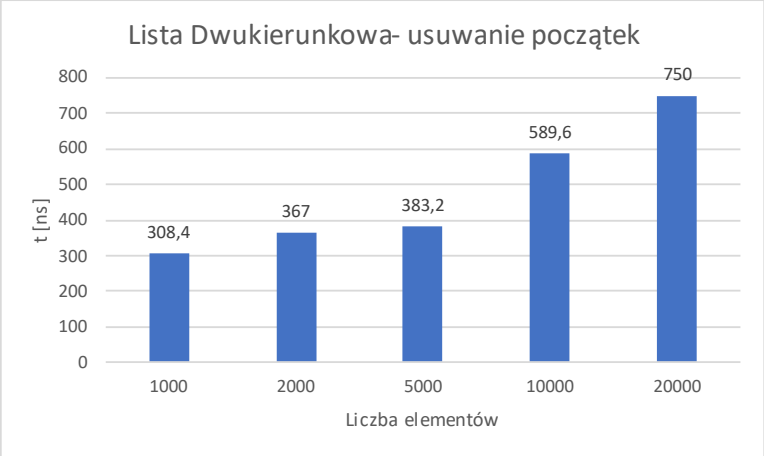
- Wybrane miejsce:



	Lista - dodawanie indeks				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	584	1000	27491	48000	71250
	583	5750	21548	30201	31000
	571	958	1620	42166	48125
	542	7833	26125	8917	38715
	582	4191	20000	13883	57541
Średni czas	572,4	3946,4	19356,8	28633,4	49326,2

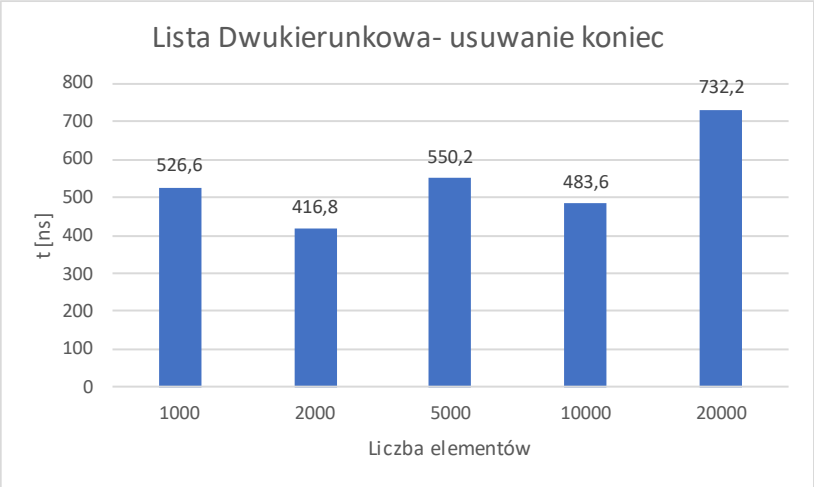
Usuwanie:

- Początek:



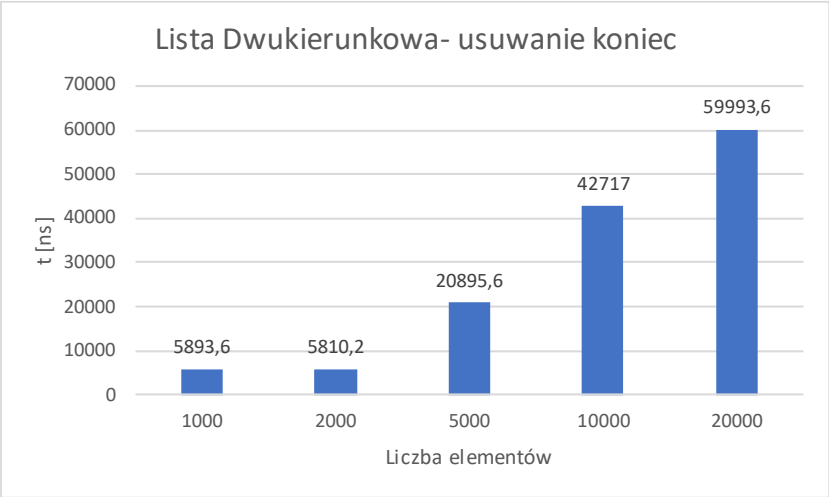
	Lista - usuwanie początek				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	292	459	292	708	709
	291	542	333	417	625
	292	417	332	667	547
	375	334	709	541	792
	292	83	250	615	458
Średni czas	308,4	367	383,2	589,6	750

- Koniec:



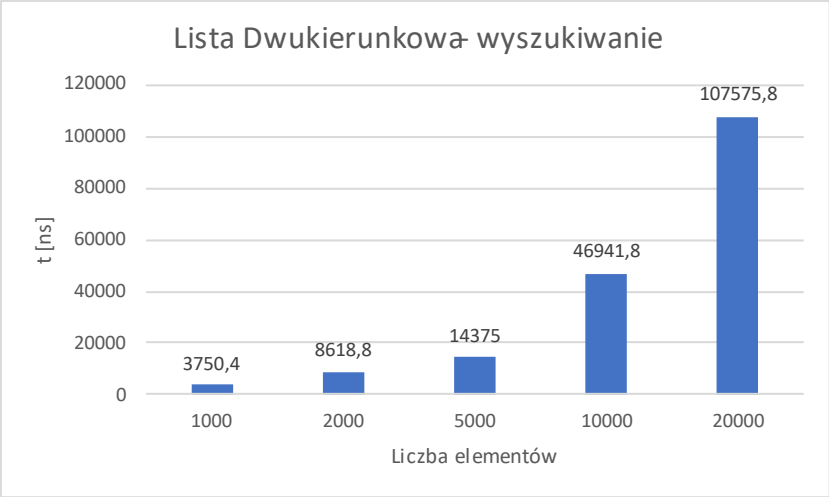
	Lista - usuwanie koniec				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	500	250	750	709	625
	875	292	417	625	792
	415	500	375	334	667
	593	250	292	417	789
	250	792	917	333	788
Średni czas	526,6	416,8	550,2	483,6	732,2

- Wybrane miejsce:



	Lista - usuwanie indeks				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	3792	12008	24152	44585	59583
	6875	4458	18750	58375	40083
	2708	2791	24865	24625	52917
	7917	7419	24836	25333	82176
	8176	2375	11875	60667	65209
Średni czas	5893,6	5810,2	20895,6	42717	59993,6

Wyszukiwanie:



	Lista - wyszukiwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	958	10417	25500	45542	224292
	2208	5000	7958	59542	20587
	6419	10333	4000	57458	59333
	6083	12756	25292	60542	228583
	3084	4588	9125	11625	5084
Średni czas	3750,4	8618,8	14375	46941,8	107575,8

Lista Dwukierunkowa (każdy element posiada wskaźnik na poprzedni i następny element listy):

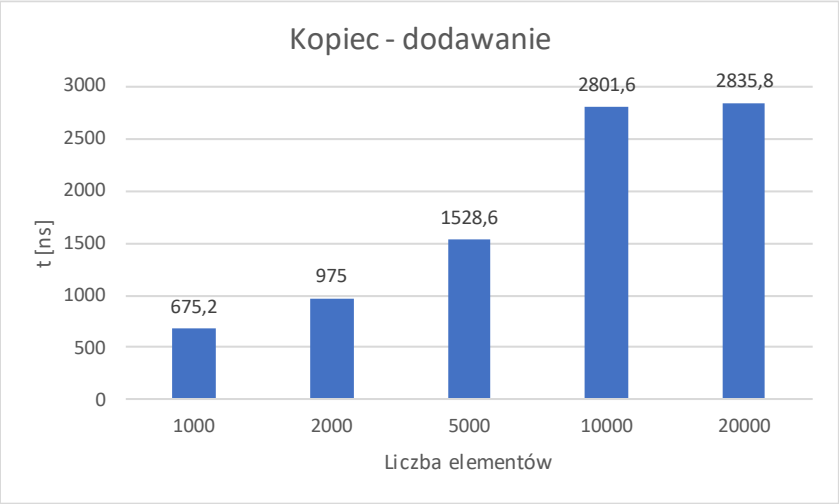
- Dodawanie elementów początek, koniec: **O(1)**
- Dodawanie wybrane miejsce **O(n)**
- Usuwanie elementów początek, koniec: **O(1)**
- Usuwanie wybrane miejsce **O(n)**
- Wyszukiwanie elementu: **O(n)**

Kopiec Binarny:

Kopiec binarny to tablicowa struktura danych reprezentująca drzewo binarne, którego wszystkie poziomy z wyjątkiem ostatniego muszą być pełne. W przypadku, gdy ostatni poziom drzewa nie jest pełny, liście ułożone są od lewej do prawej strony drzewa. Wyróżniamy dwa rodzaje kopców binarnych: kopce binarne typu max w których wartość danego węzła niebędącego korzeniem jest zawsze mniejsza niż wartość jego rodzica oraz kopce binarne typu min w których wartość danego węzła niebędącego korzeniem jest zawsze większa niż wartość jego rodzica.

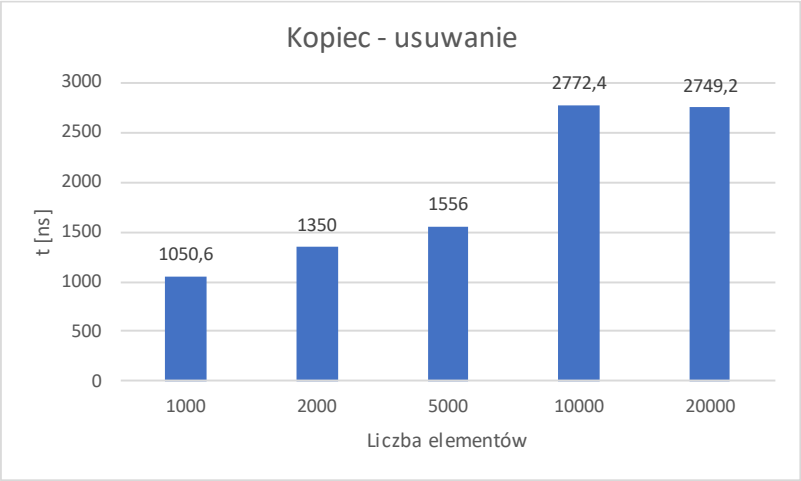
Złożoność pesymistyczna wszystkich operacji w kopcu jest równa O(ln(n)).

Dodawanie:



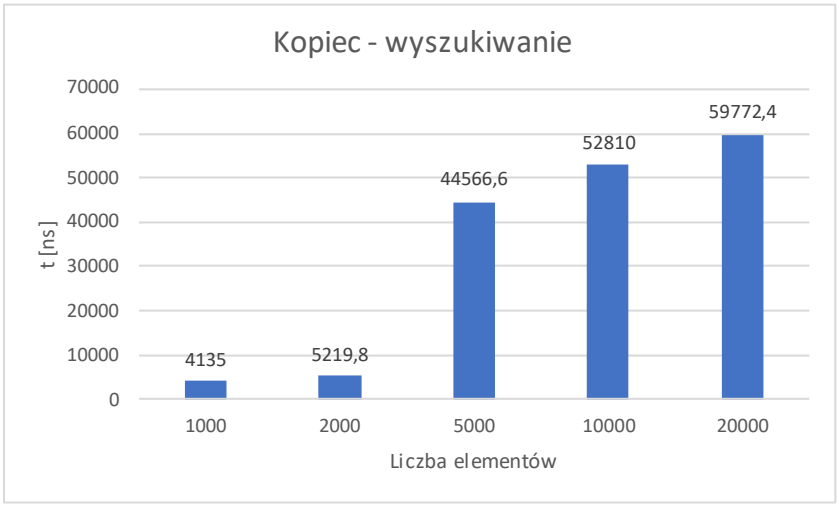
	Kopiec - dodawanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	667	750	1145	2375	2750
	792	1125	1166	3208	2971
	667	1041	1708	2976	2500
	500	1167	1791	3461	2417
	750	792	1833	1988	3541
Średni czas	675,2	975	1528,6	2801,6	2835,8

Usuwanie:



	Kopiec - usuwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	961	1000	1250	1042	3083
	875	3333	1500	4167	2593
	1625	875	1656	4000	2500
	1167	750	1666	2537	2594
	625	792	1708	2116	126
Średni czas	1050,6	1350	1556	2772,4	2179,2

Wyszukiwanie:



	Kopiec - wyszukiwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	1875	2172	25792	79375	28625
	3008	11792	54917	11625	69584
	875	4279	43166	45557	64417
	7917	3167	60208	57943	51625
	7000	4689	38750	69550	84611
Średni czas	4135	5219,8	44566,6	52810	59772,4

Kopiec Binarny:

- Dodawanie elementów: **O(ln n)**
- Usuwanie elementów: **O(ln n)**
- Wyszukiwanie elementu: **O(ln n)**

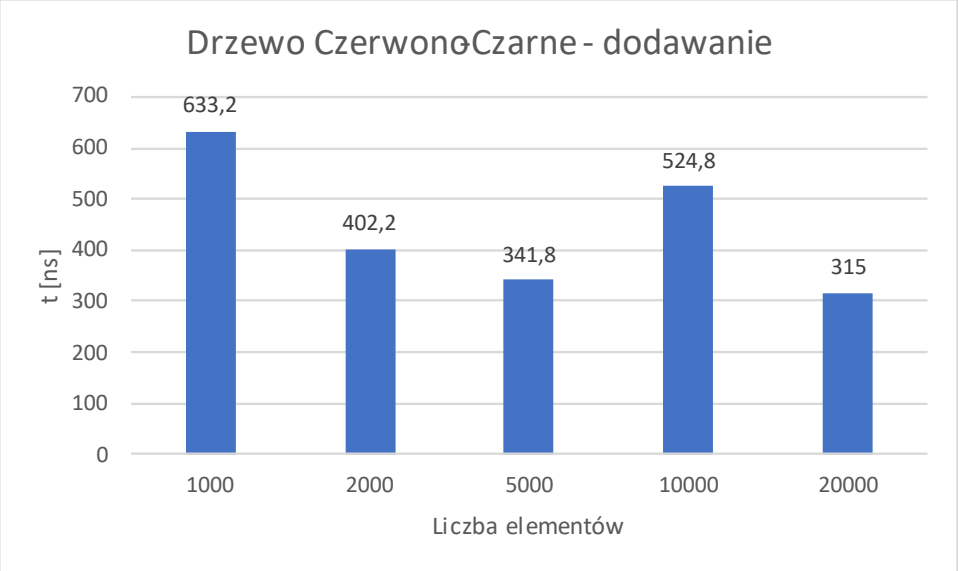
Drzewo Czerwono-Czarne:

Rodzaj samoorganizującego się binarnego drzewa poszukiwań - struktury danych stosowanej w informatyce najczęściej do implementacji tablic asocjacyjnych.

W drzewie czerwono-czarnym z każdym węzłem powiązany jest dodatkowy atrybut, *kolor*, który może być czerwony lub czarny. Oprócz podstawowych własności drzew poszukiwań binarnych, wprowadzone zostały kolejne wymagania, które trzeba spełniać:

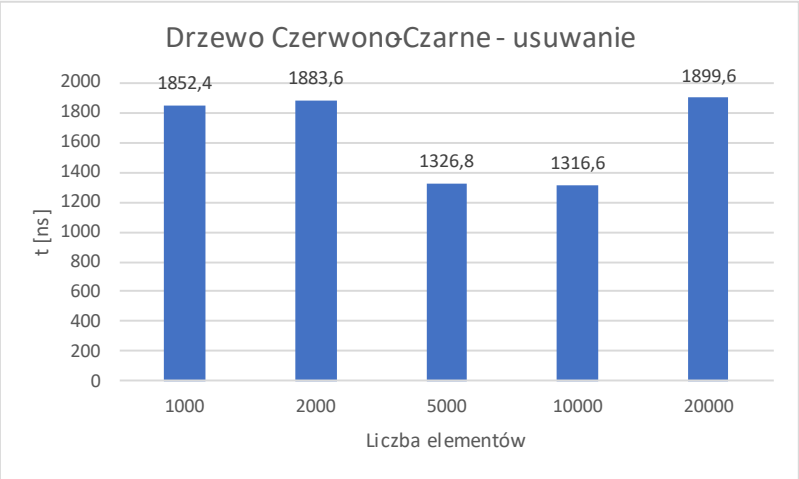
1. Każdy węzeł jest czerwony albo czarny.
2. Korzeń jest czarny.
3. Każdy liść jest czarny (Można traktować *nil* jako liść).
4. Jeśli węzeł jest czerwony, to jego synowie muszą być czarni.
5. Każda ścieżka z ustalonego węzła do każdego z jego potomków będących liśćmi liczy tyle samo czarnych węzłów.

Dodawanie:



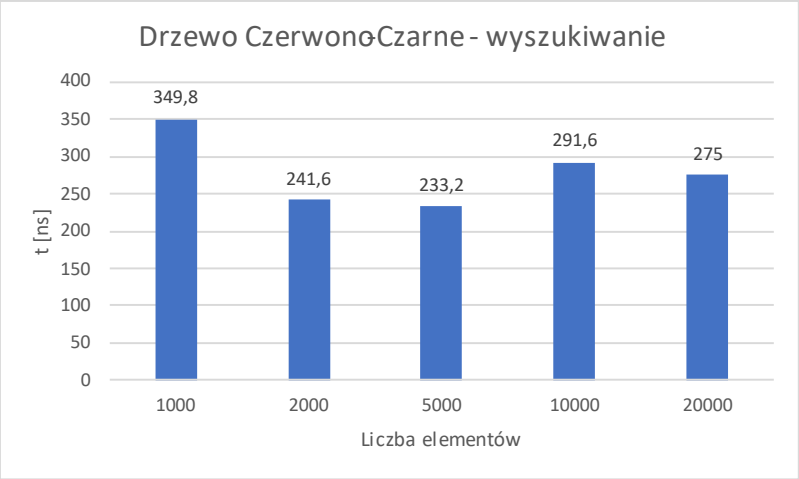
	Drzewo Czerwono-Czarne - dodawanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	1042	584	375	208	708
	750	333	584	333	325
	458	259	333	750	250
	708	292	167	1083	125
	208	543	250	250	167
Średni czas	633,2	402,2	341,8	524,8	315

Usuwanie:



	Drzewo Czerwono-Czarne - usuwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	2375	1250	2042	1500	1250
	1137	1959	1250	1125	1208
	1917	1084	1083	833	1541
	1417	2583	1093	1750	708
	2416	2542	1166	1375	4791
Średni czas	1852,4	1883,6	1326,8	1316,6	1899,6

Wyszukiwanie:



	Drzewo Czerwono-Czarne - wyszukiwanie				
Ilość pomiarów	1000	2000	5000	10000	20000
Czas w mikrosekundach [ns]	333	167	208	542	333
	250	208	250	208	250
	666	250	208	250	292
	167	250	250	250	292
	333	333	250	208	208
Średni czas	349,8	241,6	233,2	291,6	275

Drzewo Czerwono-Czarne:

- Dodawanie elementów: **O(h)**
- Usuwanie elementów: **O(h)**
- Wyszukiwanie elementu: **O(h)**

3. Wnioski:

Udało mi się zaimplementować 4 struktury danych: tablicę dynamiczną, listę dwukierunkową, kopiec binarny oraz drzewo czerwono-czarne.

Najłatwiejsza do zaimplementowania była tablica, a najtrudniejsze okazało się drzewo czerwono czarne, przez jego złożone właściwości oraz dużo różnych przypadków podczas wykonywania operacji dodawania oraz usuwania.

Przy liście złożoność dodawania i usuwania elementu na wybranym miejscu nie ma złożoności O(1), gdyż musimy najpierw iterować po liście żeby dostać się do pożądanego miejsca, co jest bardzo nieefektywne. Co za tym idzie jeśli strukturę będziemy wykorzystywać do dodawania elementu na początek lub koniec struktury to najlepsza wydaje się lista. Jeśli jednak chcemy także dodawać elementy w środek to lepsze zdecydowanie będzie drzewo czerwono-czarne. Do iteracji jednak lepsza okazuje się tablica dynamiczna, prze to że jej elementy są ułożone po kolei w pamięci.

Jeśli natomiast chcemy wyszukiwać jakiś element to do tego najlepsze będzie drzewo czerwono czarne, które dzięki właśnie swoim właściwościom najczęściej daje nam najlepszy czas. Dobre także do tego zastosowania wydają się tablica lub kopiec, ale to w przypadku gdy mamy małą strukturę.