Wrocław 24.04.19 r.

Struktury danych i złożoność obliczeniowa

Wydział Elektroniki
Kierunek: Informatyka
Nr albumu: 241390
Imię i Nazwisko: Paweł Parczyk
Grupa zajęciowa: Wtorek TP 13-15
Kod grupy: E05-00d
Prowadzący: dr inż. Zbigniew Buchalski

Temat

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

Ocena:

Data: 7.05.19 r.

Sprawozdanie z przebiegu eksperymentu

Wstęp: Projekt zakładał napisanie czterech struktur danych. Są nimi tablica dynamiczna, lista, kopiec i drzewo czerwono-czarne, a następnie sprawdzenie czasów działania poszczególnych funkcji na zaimplementowanych strukturach danych. Na tablicy i liście należało sprawdzić czasy dla operacji dodawania, usuwania i wyszukania elementów. Usuwanie i dodawanie były rozdzielone na 3 różne procedury: początku, z końca i ze środka struktury. Na kopcu i drzewie należało przeprowadzić funkcje dodawania na początek i wyszukiwanie elementu.

1. **Tablica dynamiczna:**
   a) Dodawanie elementu: złożoność tej operacji wynosi O(1), gdy tablica może zostać rozszerzona bez potrzeby relokacji pamięci. Zrealizowana tablica miała zajmować jak najmniej miejsca więc za każdym dodaniem trzeba relokować pamięć co zmienia klasę złożoności do O(n).
   b) Usuwanie elementu: złożoność wynosi O(n) gdyż trzeba przesunąć wszystkie elementy.
   c) Wyszukanie elementu: złożoność wynosi O(n) gdyż trzeba przeiterować po wszystkich elementach**.**
2. **Lista jednokierunkowa:**
   a) Dodawanie elementu: złożoność klasy O(1) gdy wstawiany element ma się znaleźć na początku, w przeciwnym wypadu złożoność wynosi O(n) ponieważ trzeba przeiterować w poszukiwaniu właściwej pozycji, skrajnym przypadkiem wstawiania w środku jest wstawienie elementu na końcu.
   b) Usuwanie elementu: podobnie jak przy dodawaniu, usuwanie z początku ma złożoność O(1) a każde inne złożoność O(n).
   c) Wyszukiwanie elementu: O(n) wyszukanie oznacza przejście listy aż do szukanego elementu.
3. **Kopiec zaimplementowany z wykorzystaniem tablicy**
   a) Dodawanie elementu: złożoność O(n log n). Dodanie nowej wartości wymaga sprawdzenia warunku kopca
   b) Usunięcie elementu: złożoność O(n log n).
   c) Wyszukanie elementu: O(n) w przypadku implementacji na tablicy wyszukiwać można tak samo jak w tablicy
4. **Drzewo czerwono-czarne**
   a) Dodawanie elementu: O(h) zależy od głębokości drzewa.
   b) Usuwanie elementu: podobnie jak przy dodawaniu złożoność obliczeniowa usuwanie wynosi O(h) zależy od głębokości drzewa.

c) Wyszukiwanie elementu: w drzewie zrównoważonym jakim jest drzewo czerwono-czarne wyszukiwanie ma złożoność O(log(2) n), ponieważ zależy od głębokości drzewa, które jest możliwe najmniejsze.

Plan eksperymentu: Eksperyment składał się z 6 testów z populacjami różnej wielkości. W każdym teście zostało wygenerowanych 100 populacji o jednakowej wielkości, następnie stworzono nowe struktury przeprowadzono test wyszukiwania i usunięto elementy z struktur.

Otrzymane wyniki czasowe zostały uśrednione w celu usunięcie błędów losowych.
Do przeprowadzenia eksperymentu zostały wybrane populacje o wielkości 100, 200, 500, 1000, 2000 i 5000 elementów. Liczby te zostały dobrane w oparciu o możliwości maszyny, na której były przeprowadzane testy.

Program testujący został napisany w języku c++ z wykorzystaniem technik programowania obiektowego, następnie został skompilowany z użyciem kompilatora g++. Testy zostały przeprowadzone na komputerze zarządzanym przez system *Linux Ubuntu 16.04*. Do pomiaru czasów realizacji funkcji została użyta linuxowa funkcja *gettimeofday z biblioteki <sys/time.h>*, której rozdzielczość wynosi jedna milisekunda.

Eksperyment

Poniżej zaprezentowane zostały czasy wykonania poszczególnych operacji na strukturach. Pomiar każdej z funkcji składa się z dwóch tabel. Pierwsza prezentuje sumaryczne czasy poszczególnych operacji. Tzn: komórka w pierwszej tabeli [dodaj (koniec) x 100] zawiera informację ile trwało dodanie 100 elementów na koniec tablicy dynamicznej. Druga tabela zawiera średni czas pojedynczego wykonania operacji tzn: komórka [dodaj (koniec) x 100] zawiera informację ile średnio trwało dodanie elementu do tablicy dynamicznej.

Tablica dynamiczna:

Sumaryczne czasy operacji dla całych struktur							
ilość danych	Dodaj [micro s] (koniec)	Usuń [micro s] (koniec)	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Dodaj [micro s] (wybrane miejsce)	Usuń [micro s] (wybrane miejsce)	Wyszukaj [micro s]
1000	3474	3427	3085	2700	2775	2425	5
2000	8686	9029	9128	8937	9940	8818	9
5000	54487	55427	55407	54651	59097	54452	23
10000	224361	225379	223223	222626	241203	228757	46
20000	1061010	1092030	993779	1064020	1177994	1125008	119
50000	5993930	5976760	6016149	6298418	6777101	6101354	223

Średni czas realizacji pojedynczej funkcji dla przebiegów w różnych populacjach							
ilość danych	Dodaj [micro s] (koniec)	Usuń [micro s] (koniec)	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Dodaj [micro s] (wybrane miejsce)	Usuń [micro s] (wybrane miejsce)	Wyszukaj [micro s]
1000	3,474	3,427	3,085	2,7	2,775	2,425	0,005
2000	4,343	4,5145	4,564	4,4685	4,97	4,409	0,0045
5000	10,8974	11,0854	11,0814	10,9302	11,8194	10,8904	0,0046
10000	22,4361	22,5379	22,3223	22,2626	24,1203	22,8757	0,0046
20000	53,0505	54,6015	49,68895	53,201	58,8997	56,2504	0,00595
50000	119,8786	119,5352	120,32298	125,96836	135,54202	122,02708	0,00446

Lista jednokierunkowa:

Sumaryczne czasy operacji dla całych struktur							
ilość danych	Dodaj [micro s] (koniec)	Usuń [micro s] (koniec)	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Dodaj [micro s] (wybrane miejsce)	Usuń [micro s] (wybrane miejsce)	Wyszukaj [micro s]
1000	2818	2356	26	30	1807	963	7
2000	14618	14231	50	63	9632	5243	12
5000	111835	108554	128	169	75363	49560	32
10000	661179	646051	337	414	446358	281033	72
20000	3983571	3608277	765	1027	2621249	1608724	139
50000	40437654	35380480	2903	3913	24335201	14150058	324

Średni czas realizacji pojedynczej funkcji dla przebiegów w różnych populacjach							
ilość danych	Dodaj [micro s] (koniec)	Usuń [micro s] (koniec)	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Dodaj [micro s] (wybrane miejsce)	Usuń [micro s] (wybrane miejsce)	Wyszukaj [micro s]
1000	2,818	2,356	0,026	0,03	1,807	0,963	0,007
2000	7,309	7,1155	0,025	0,0315	4,816	2,6215	0,006
5000	22,367	21,7108	0,0256	0,0338	15,0726	9,912	0,0064
10000	66,1179	64,6051	0,0337	0,0414	44,6358	28,1033	0,0072
20000	199,17855	180,41385	0,03825	0,05135	131,06245	80,4362	0,00695
50000	808,75308	707,6096	0,05806	0,07826	486,70402	283,00116	0,00648

Kopiec:

Sumaryczne czasy operacji dla całych struktur			
ilość danych	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Wyszukaj [micro s]
1000	231	16797	667
2000	495	69448	1487
5000	1315	459342	4323
10000	2898	1932835	9855
20000	5637	7484912	20143
50000	15210	46842589	62353

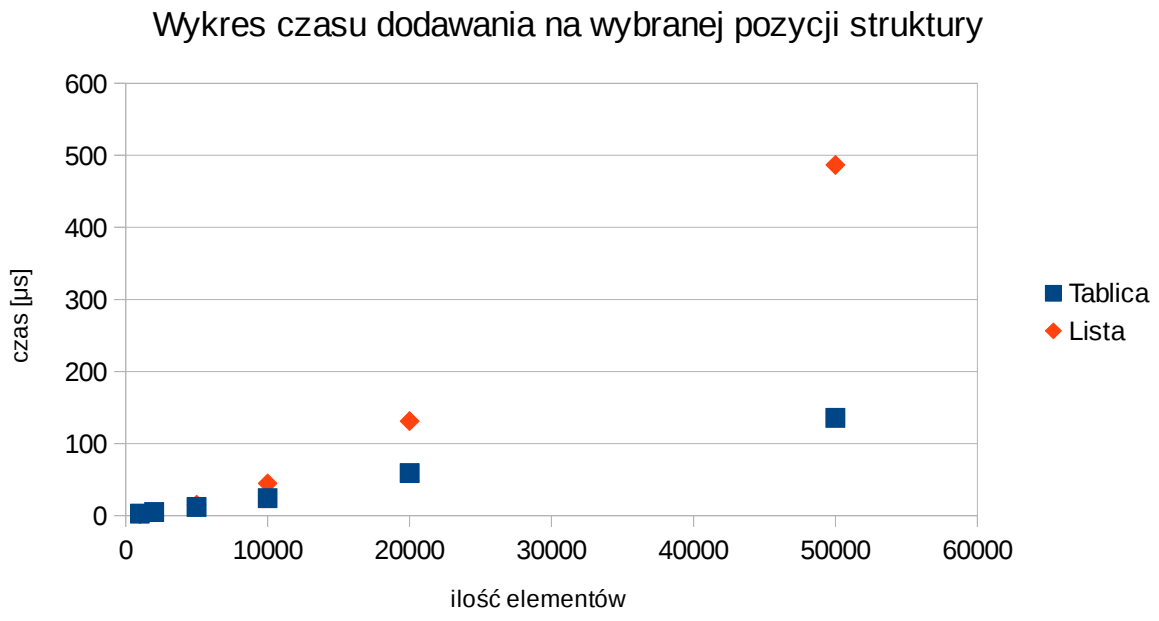
Średni czas realizacji pojedynczej funkcji			
ilość danych	Dodaj [micro s] (początek)	Usuń [micro s] (początek)	Wyszukaj [micro s]
1000	0,231	16,797	0,667
2000	0,2475	34,724	0,7435
5000	0,263	91,8684	0,8646
10000	0,2898	193,2835	0,9855
20000	0,28185	374,2456	1,00715
50000	0,3042	936,85178	1,24706

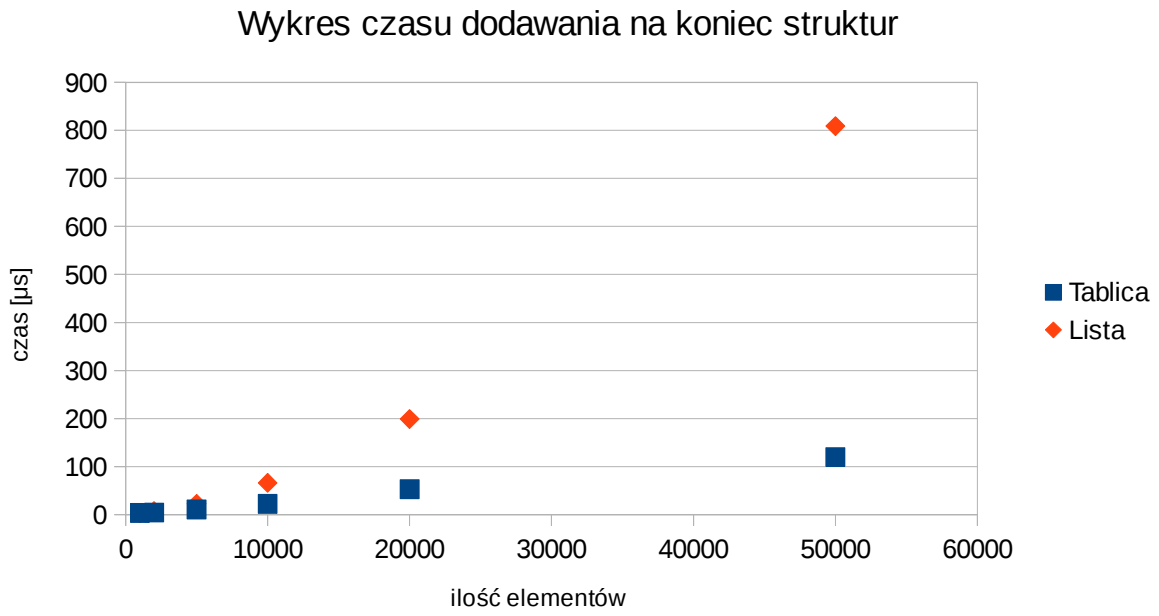
Drzewo czerwono-czarne:

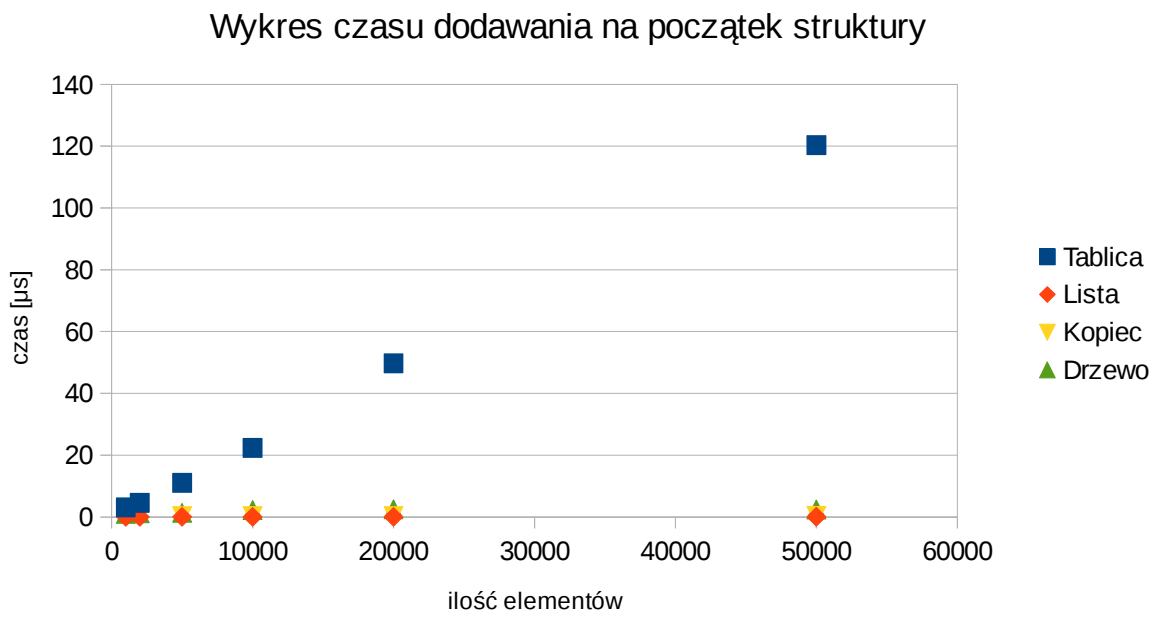
Sumaryczne czasy operacji dla całych struktur		
ilość danych	Dodaj [micro s] (początek)	Wyszukaj [micro s]
1000	1038	727
2000	2344	1709
5000	6646	5045
10000	22605	16166
20000	50352	45251
50000	121463	108969

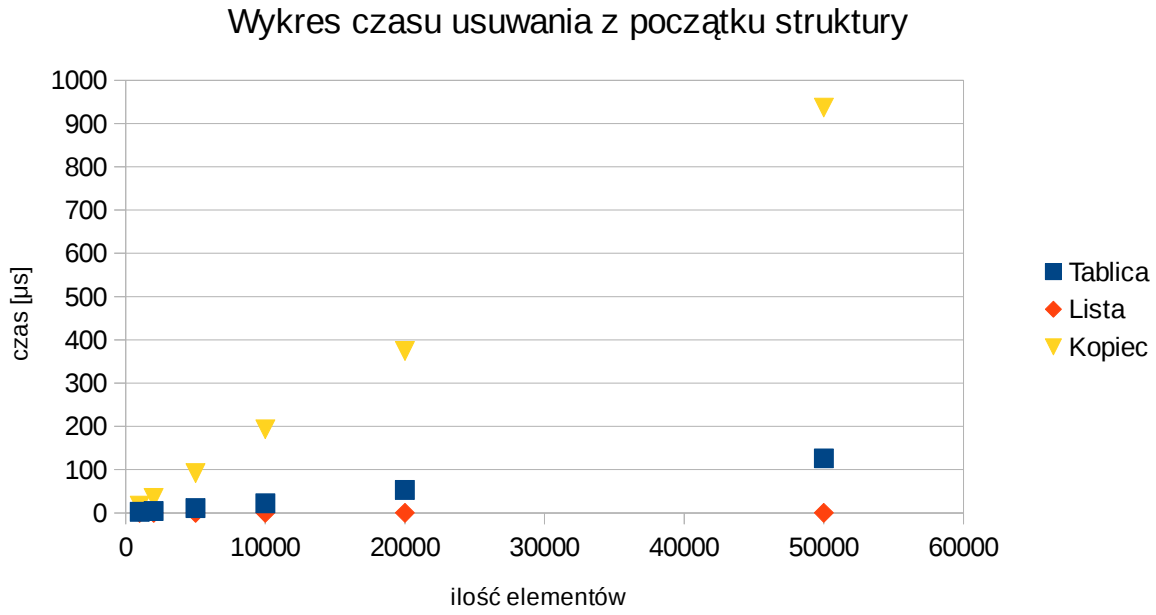
Średni czas realizacji pojedynczej funkcji		
ilość danych	Dodaj [micro s] (początek)	Wyszukaj [micro s]
1000	1,038	0,727
2000	1,172	0,8545
5000	1,3292	1,009
10000	2,2605	1,6166
20000	2,5176	2,26255
50000	2,42926	2,17938

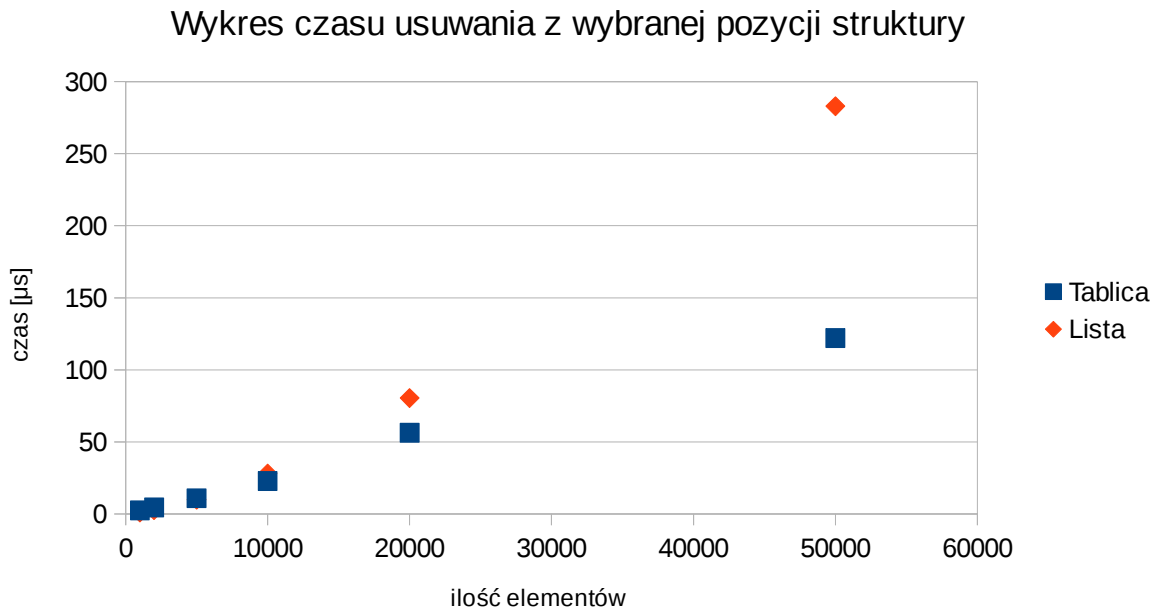
Z uzyskanych danych wygenerowano wykresy w celu oszacowania faktycznej złożoności obliczeniowej zaimplementowanych operacji na strukturach danych.

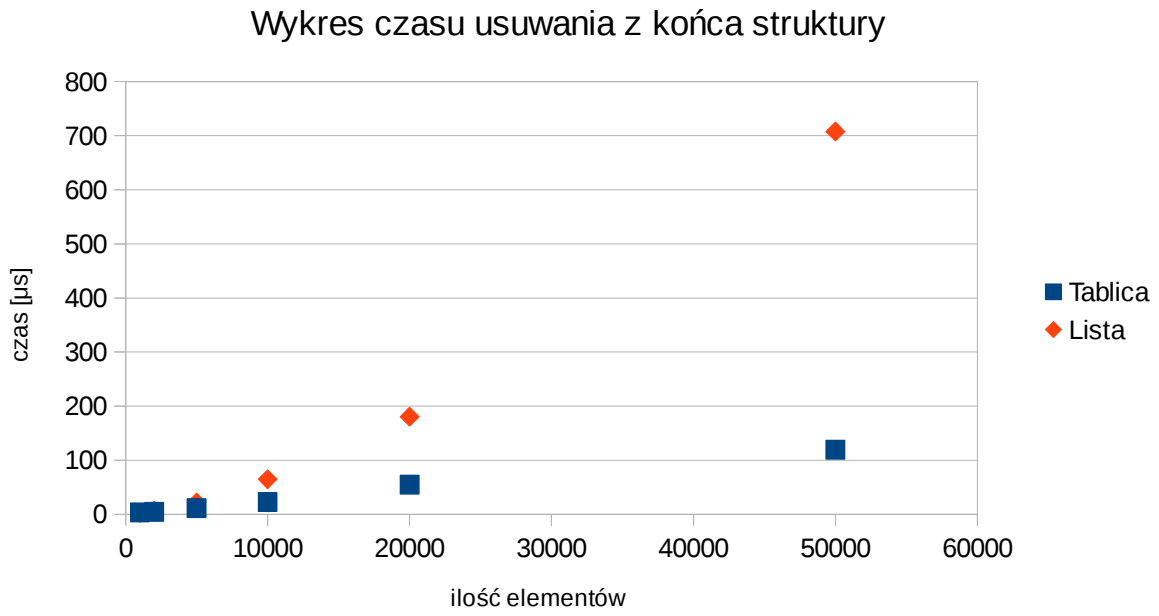


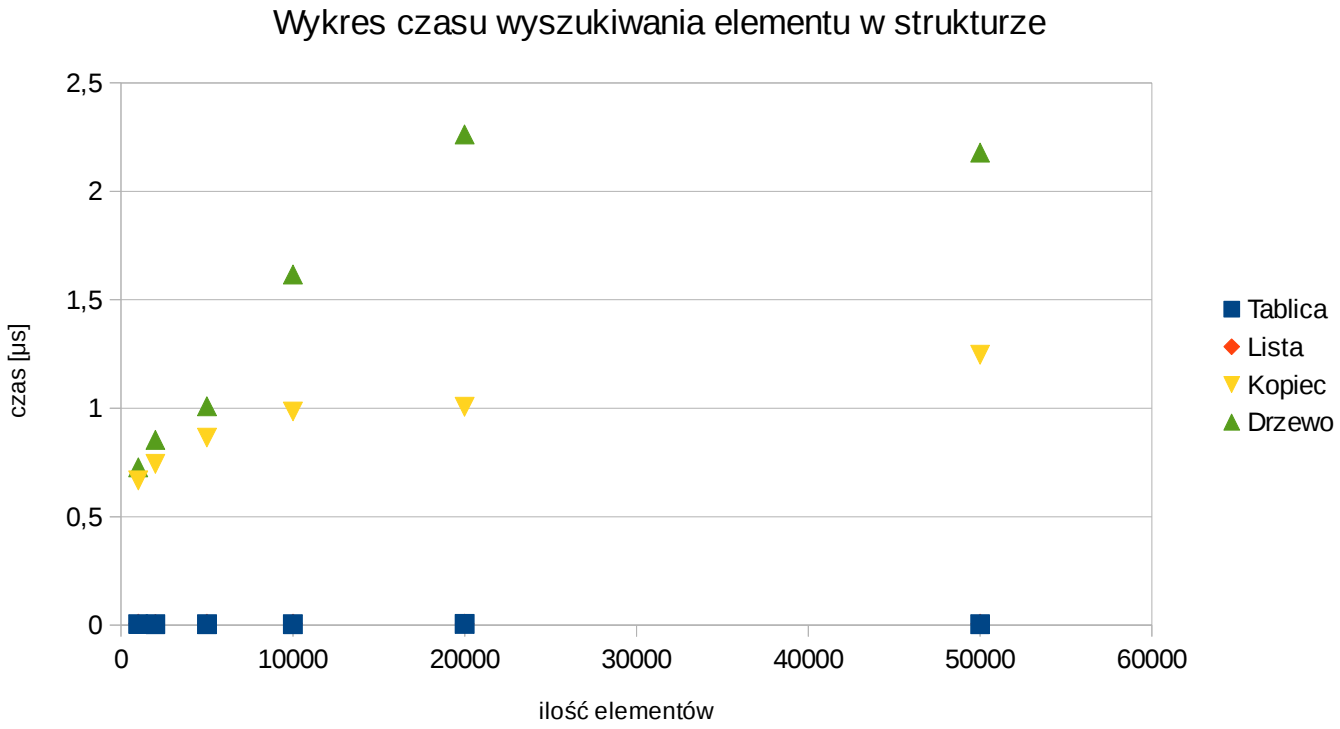

Wykres czasu dodawania na koniec struktur
900
800
700
600
500
400
300
200
100
0
czas [µs]
0
10000
20000
30000
40000
50000
60000
ilość elementów
Tablica
Lista


Wykres czasu dodawania na początek struktury
140
120
100
80
60
40
20
0
czas [µs]
0
10000
20000
30000
40000
50000
60000
ilość elementów
Tablica
Lista
Kopiec
Drzewo


Wykres czasu usuwania z początku struktury
1000
900
800
700
600
500
400
300
200
100
0
czas [µs]
0
10000
20000
30000
40000
50000
60000
ilość elementów
Tablica
Lista
Kopiec


Wykres czasu usuwania z wybranej pozycji struktury
300
250
200
150
100
50
0
czas [µs]
0
10000
20000
30000
40000
50000
60000
ilość elementów
Tablica
Lista





Wnioski: Złożoność obliczeniowa zaimplementowanych struktur nie odbiega od złożoności obliczeniowych podanych w literaturze.

Z uwagi na różne cechy struktur każda znajduje zastosowanie przy różnych operacjach i działaniach. W zależności od charakteru przetwarzanych danych stosuje się inne struktury.

Zalety i wady:

Tablica: jest to struktura która charakteryzuje się bardzo szybkim dostępem do elementu, za to bardzo źle znosi modyfikację danych wewnątrz. Zajmuje jej to bardzo dużo czasu. Struktura ta mogłaby znaleźć zastosowanie wszędzie tam gdzie nie ma częstych modyfikacji ilości elementów.

Lista: lista jest strukturą idealną gdy często jest modyfikowana populacja, ale nie same wartości. Dodanie lub usunięcie danych odbywa się w bardzo szybkim czasie. Za to wyszukanie już jest znacznie wolniejsze od tablicy

Drzewo czerwono-czarne: drzewo czerwono czarne jest drzewem samo balansującym się co powoduje niezwykle dobre złożoności obliczeniowe wyszukiwania. Może byś stosowane wszędzie tam gdzie iterowanie po tablicy czy liście daje niewystarczająco szybkie rezultaty.

Podczas eksperymentów zaobserwowano bardzo niespodziewane czasy wyszukiwania w strukturach drzewiastych. Z wykresów widać że zaimplementowane algorytmy zachowały pożądane złożoności obliczeniowe, ale czasy ich wykonania nawet dla bardzo małych populacji są niespodziewanie długie. Może to być spowodowane architekturą realizacji pracy równoległej. Funkcje wyszukujące w drzewie i kopcu zostały zaimplementowane jako funkcje rekurencyjne. Istnieje możliwość ze podczas wywołania kolejnych iteracji system przeskakiwał do innych zadań co spowodowało przestoje.

Kopiec: jako struktura która na szczycie zawiera największy element idealnie nadaje się do realizacji kolejki priorytetowej.

W eksperymentach udało się uzyskać lepszą złożoność obliczeniową niż znaleziona jako wzorcowa. W algorytmie wykorzystano przeglądanie drzewa w głąb z odrzucaniem poddrzew w których korzeń jest mniejszy od elementu wyszukiwanego wykorzystując cechę drzewa, że każde dziecko jest mniejsze od swojego rodzica. Na wykresach widać, ze drzewo ma lepszą złożoność od kopca więc nie jest to rozwiązanie idealne i tak lepsze od liniowego przeszukiwania tablicy.