STRUKTURY DANYCH I ZŁOŻONOŚĆ OBLICZENIOWA – PROJEKT

Temat: Badanie efektywności operacji na danych w podstawowych strukturach danych.

Autor: Michał Tomaszewicz, 235568

Prowadzący projekt: Dr inż. Dariusz Banasiak

1.Wstęp

W projekcie zajmowałem się trzema podstawowymi strukturami danych. Tablicą, listą dwukierunkową oraz kopcem. Wszystkie te struktury, wraz z niezbędnymi metodami, zaimplementowałem w programie w języku C++. Celem projektu było zbadanie efektywności operacji przeprowadzanych na tych strukturach z zależności od ich wielkości. Do operacji należały: dodanie elementu, usunięcie elementu oraz wyszukanie elementu. Dodatkowym warunkiem (w przypadku listy i tablicy) było wyróżnienie operacji dodawanie i usuwania elementu na trzy warianty dotyczące miejsca w strukturze – operacja na początku, na końcu oraz w środku. Element środkowy był losowany spośród wszystkich możliwych miejsc w strukturze.

Teoretyczne złożoności obliczeniowe w przypadku średnim:

Tablica:

- dodanie elementu O(n)
- usunięcie elementu O(n)
- wyszukanie elementu O(n)

Lista:

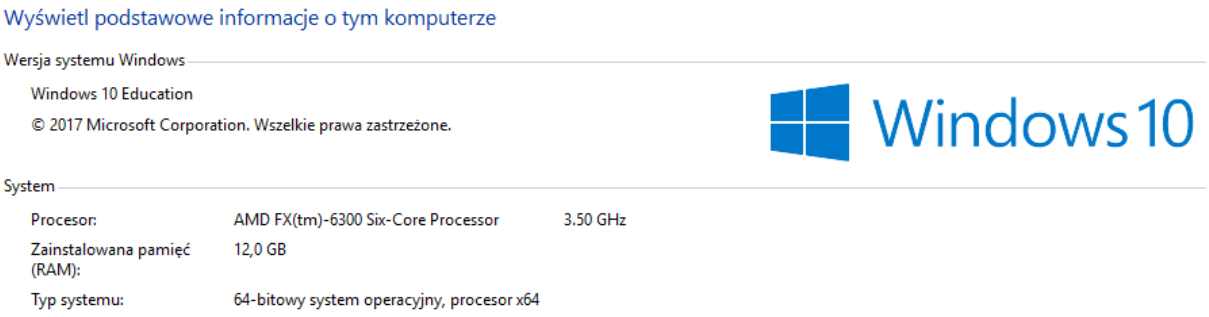
- dodanie elementu O(n)
- usunięcie elementu O(n)
- wyszukanie elementu O(n)

Kopiec:

- dodanie elementu O(log n)
- usuwanie elementu O(log n)
- wyszukanie elementu O(n)

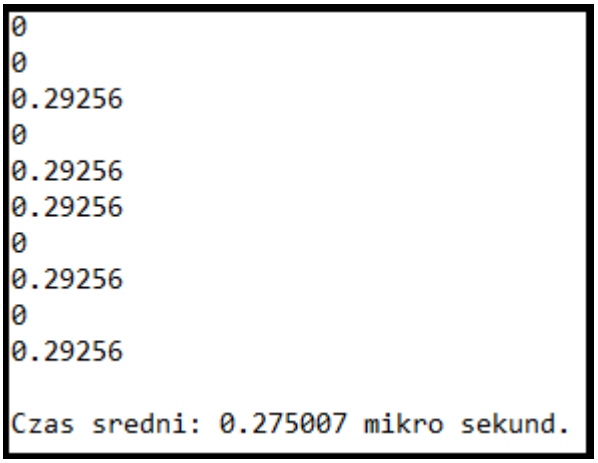
2. Plan eksperymentu

Wszystkie struktury były alokowane dynamicznie i zajmowały możliwie najmniej miejsca. Badanie efektywności badałem mierząc czasy poszczególnych operacji (czasy podawałem w mikrosekundach μs). Pomiaru czasu dokonywałem 100 razy na każdej strukturze. Wynik uśredniałem. Rozpatrywałem struktury danych 4 rozmiarów: 2000, 5000, 10000 i 20000 elementów. Elementy w strukturze generowane były przy pomocy funkcji rand(). Pomiarów dokonywałem za pomocą zaproponowanej przez Prowadzącego funkcji QueryPerformanceCounter. Pojedynczy pomiar dotyczył jedynie czasu wykonywanej operacji – nie zawiera czasu generacji danych, ani innych podobnych czynności. Do przeprowadzenia eksperymentu używałem komputera PC z procesorem AMD FX – 6300 3,50 GHz i 12 GB pamięci RAM.



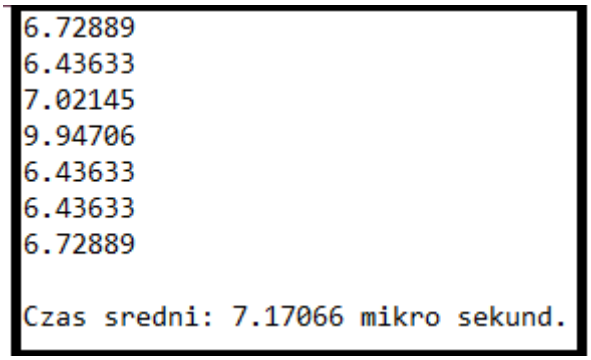
3.Wstępne obserwacje.

Po skompilowaniu programu do wersji Release i odpaleniu trybu generowania wyników eksperymentu powstały, zgodnie z planem pliki tekstowe. Po ich otworzeniu z nich okazało, że niektóre operacje wykonywane były w zerowym czasie. Jest to oczywiście niemożliwe. Wspomnę też, że w tym momencie moją podstawową jednostką czasu były mikrosekundy (ms). Dla porównania wygenerowałem wyniki w pod kontrolą środowiska Visual Studio. W tym przypadku wszystkie wyniki wyglądały na poprawne, to znaczy były większe niż zero.



Na podstawie tych obserwacji można dojść do wniosku, że środowisko Visual Studio, poprzez swoją funkcjonalność spowalnia działanie programu. Po porównaniu wyników z wersji Release i Debug (w Visual Studio) okazało się, że operacje mogą wykonywać się nawet kilkukrotnie dłużej w drugim przypadku. Spróbowałem rozwiązać ten problem - w funkcji dokonującej pomiaru czasu pomnożyć przez 1000, a następnie przez 1000’000 otrzymując wyniki w mikro i nano

sekundach. Niestety bez skutku. Dlatego wyniki, które przedstawię będą pochodziły z wersji Debug.



Skróty, których użyłem beg, end mid oznaczają po kolei operację na początku, końcu i na jednym ze środkowych elementów.

4.Wyniki dla kopca

Ilość elementów	Czas [µs]
kopiec_pop_2000	9,634
_5000	21,237
_10000	39,256
_20000	80,440

Ilość elementów	Czas [µs]
kopiec_push_2000	8,241
_5000	21,805
_10000	41,775
_20000	86,197

Ilość elementów	Czas [µs]
kopiec_search_2000	7,171
_5000	15,041
_10000	28,958
_20000	53,606

5.Wyniki dla listy

Ilość elementów	Czas [µs]
lista_pop_beg_2000	1,167
_5000	1,445
_10000	1,855
_20000	2,183

Ilość elementów	Czas [µs]
lista_pop_end_2000	0,968
_5000	1,714
_10000	2,487
_20000	3,976

Ilość elementów	Czas [µs]
lista_pop_mid_2000	10,898
_5000	28,291
_10000	74,887
_20000	225,719

Ilość elementów	Czas [µs]
lista_push_beg_2000	0,603
_5000	0,699
_10000	0,875
_20000	1,284

Ilość elementów	Czas [µs]
lista_push_end_2000	0,483
_5000	0,769
_10000	0,734
_20000	1,545

Ilość elementów	Czas [µs]
lista_push_mid_2000	9,403
_5000	20,746
_10000	43,732
_20000	92,601

Ilość elementów	Czas [µs]
lista_search_2000	21,544
_5000	53,837
_10000	119,944
_20000	205,614

5.Wyniki dla tablicy

Ilość elementów	Czas [µs]
tablica_pop_beg_2000	1,167
_5000	1,445

_10000	1,855
_20000	2,183

Ilość elementów	Czas [µs]
tablica_pop_end_2000	8,774
_5000	26,017
_10000	40,997
_20000	88,608

Ilość elementów	Czas [µs]
tablica_pop_mid_2000	10,447
_5000	23,680
_10000	53,422
_20000	85,296

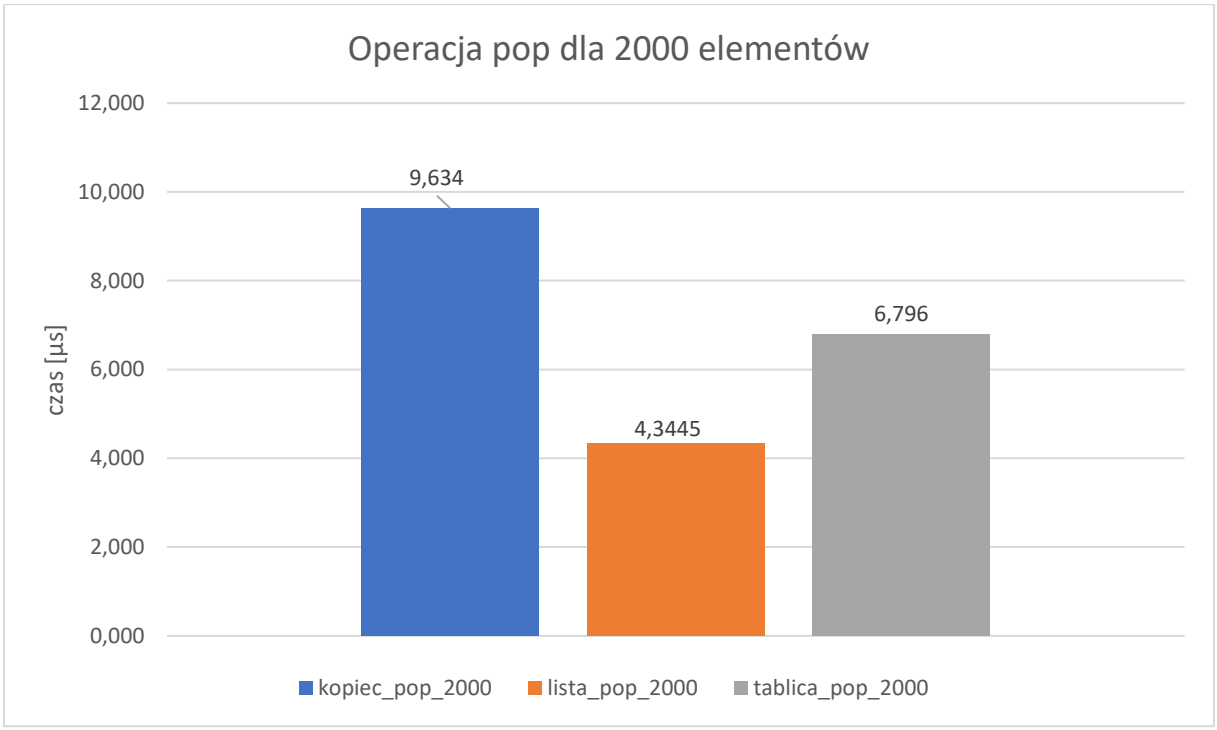
Ilość elementów	Czas [µs]
tablica_push_beg_2000	10,439
_5000	27,445
_10000	42,287
_20000	82,988

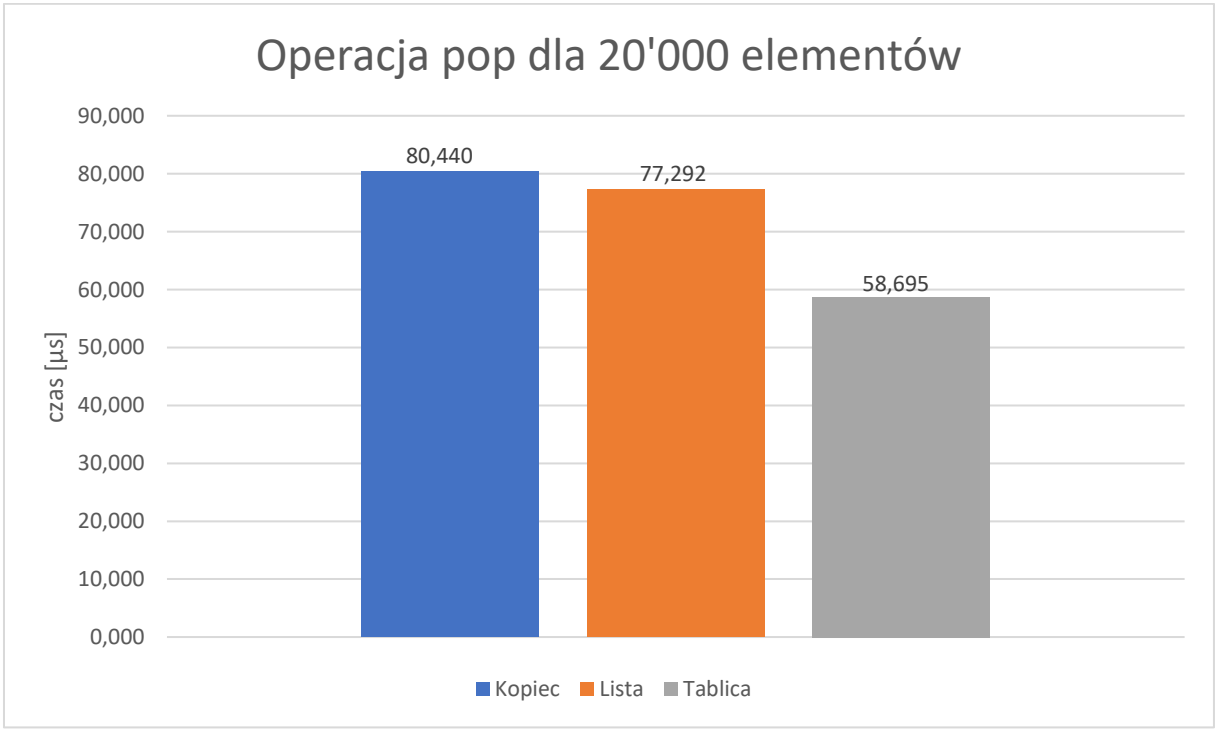
Ilość elementów	Czas [µs]
tablica_push_end_2000	13,405
_5000	24,446
_10000	42,372
_20000	103,043

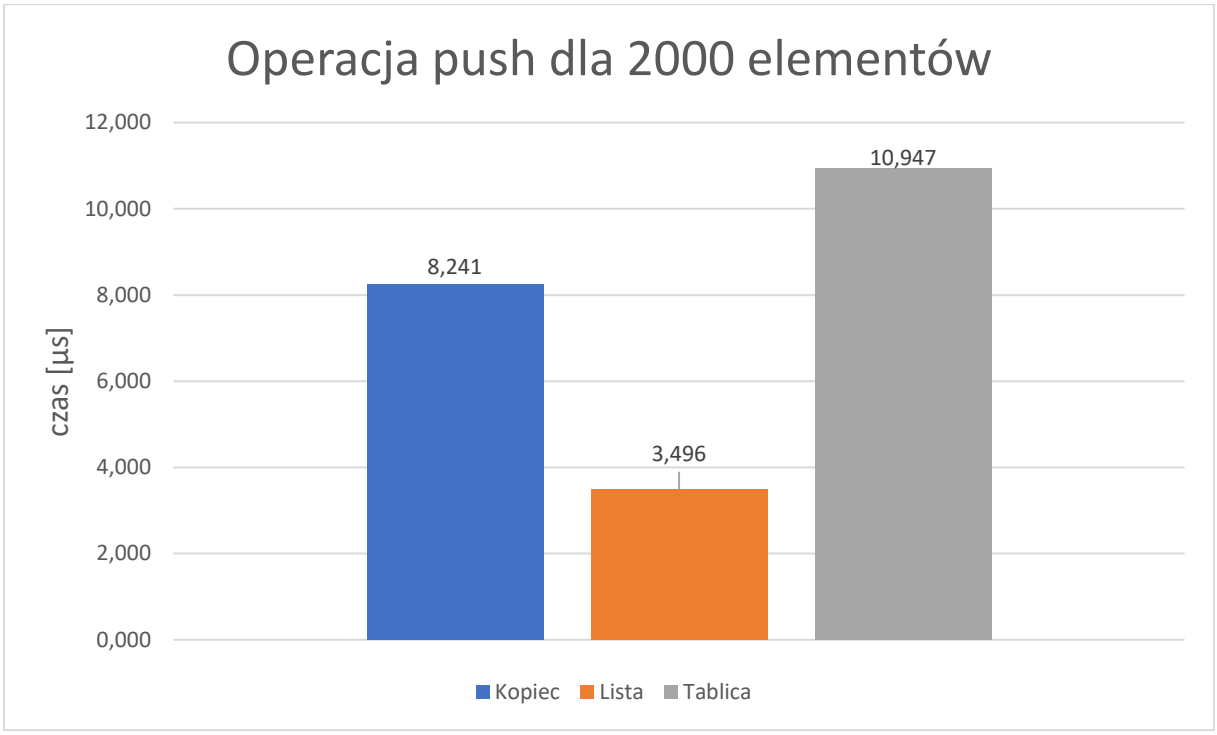
Ilość elementów	Czas [µs]
tablica_push_mid_2000	8,996
_5000	27,392
_10000	48,869
_20000	89,322

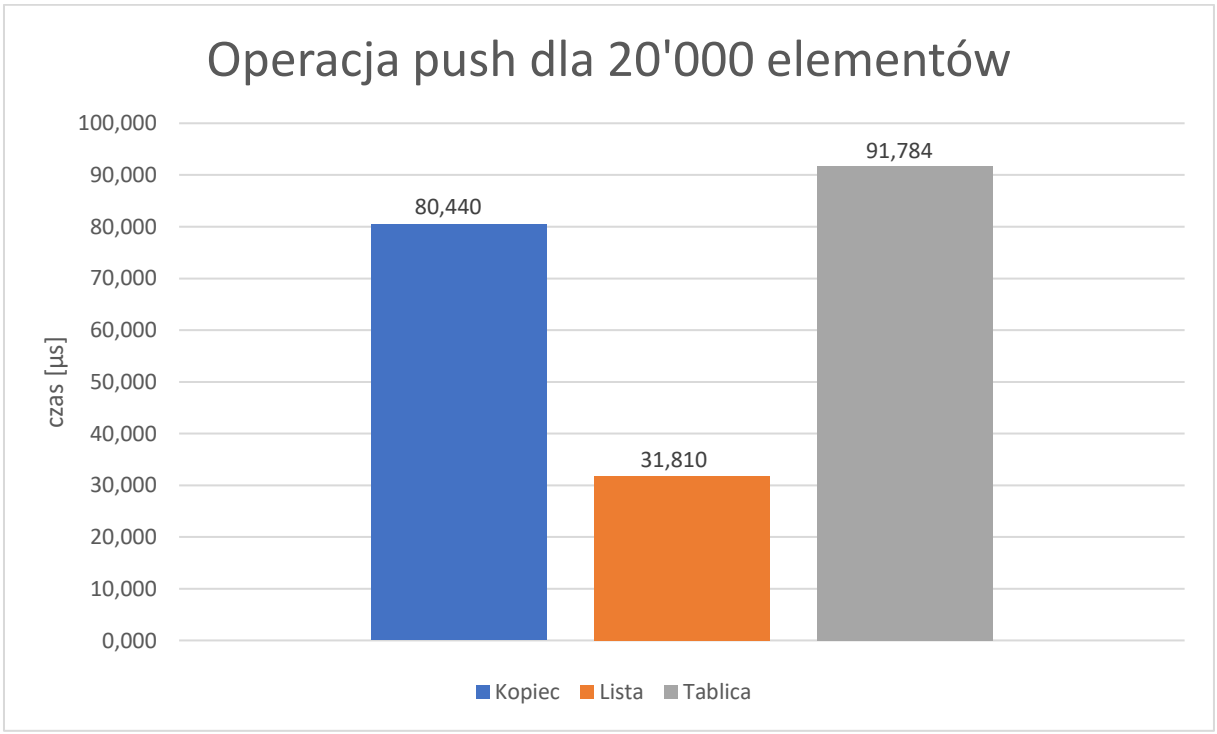
Ilość elementów	Czas [µs]
tablica_search_2000	5,889
_5000	14,005
_10000	27,647
_20000	46,766

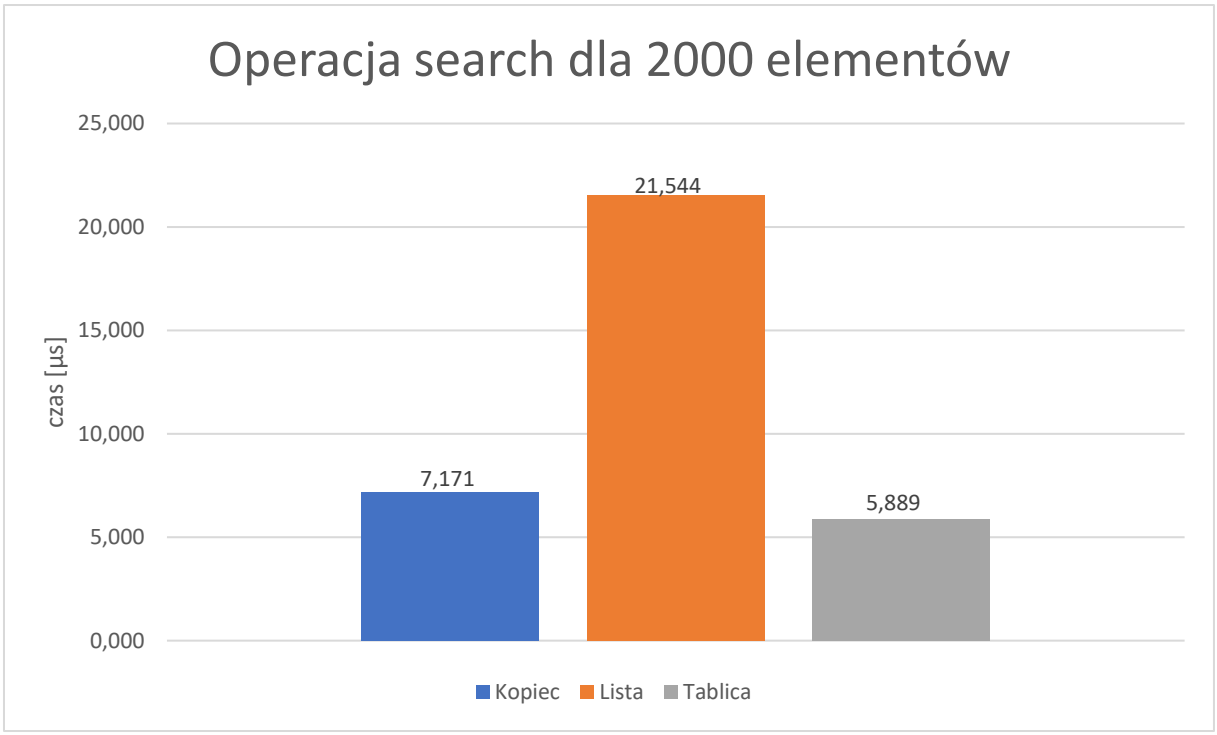
6. Wykresy

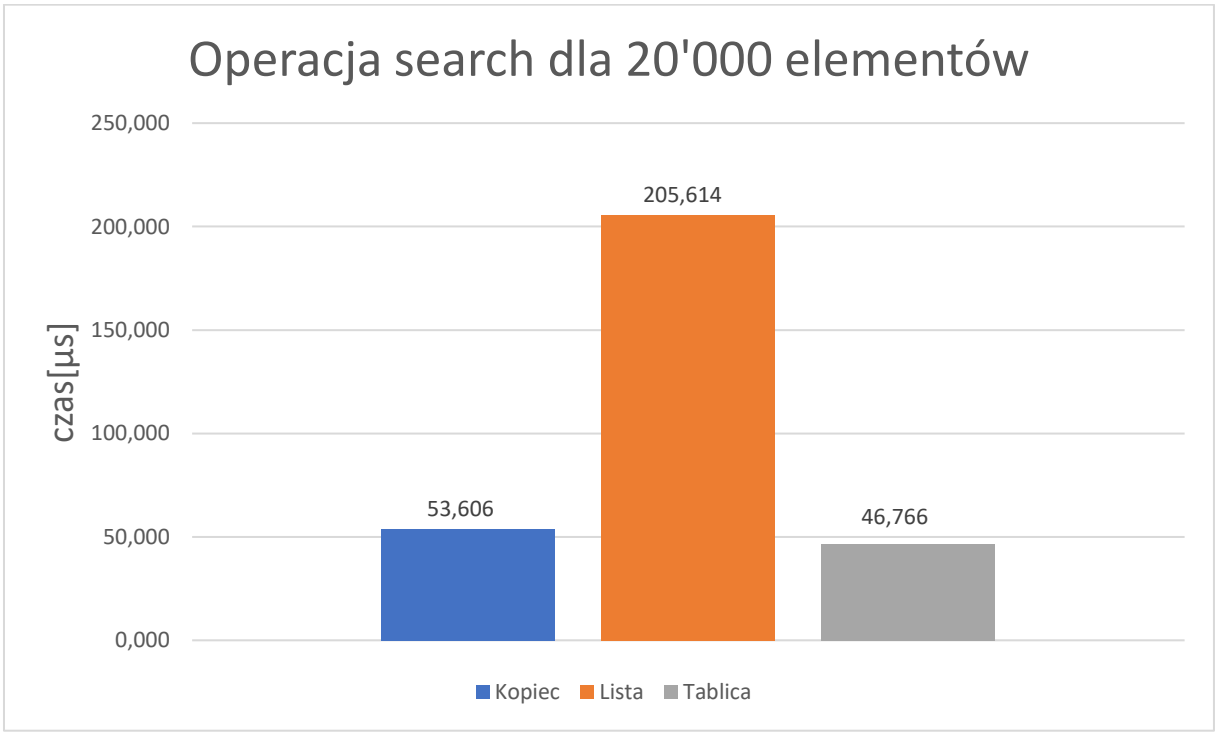

Operacja pop dla 2000 elementów
12,000
10,000
8,000
6,000
4,000
2,000
0,000
czas [µs]
9,634
4,3445
6,796
kopiec_pop_2000
lista_pop_2000
tablica_pop_2000

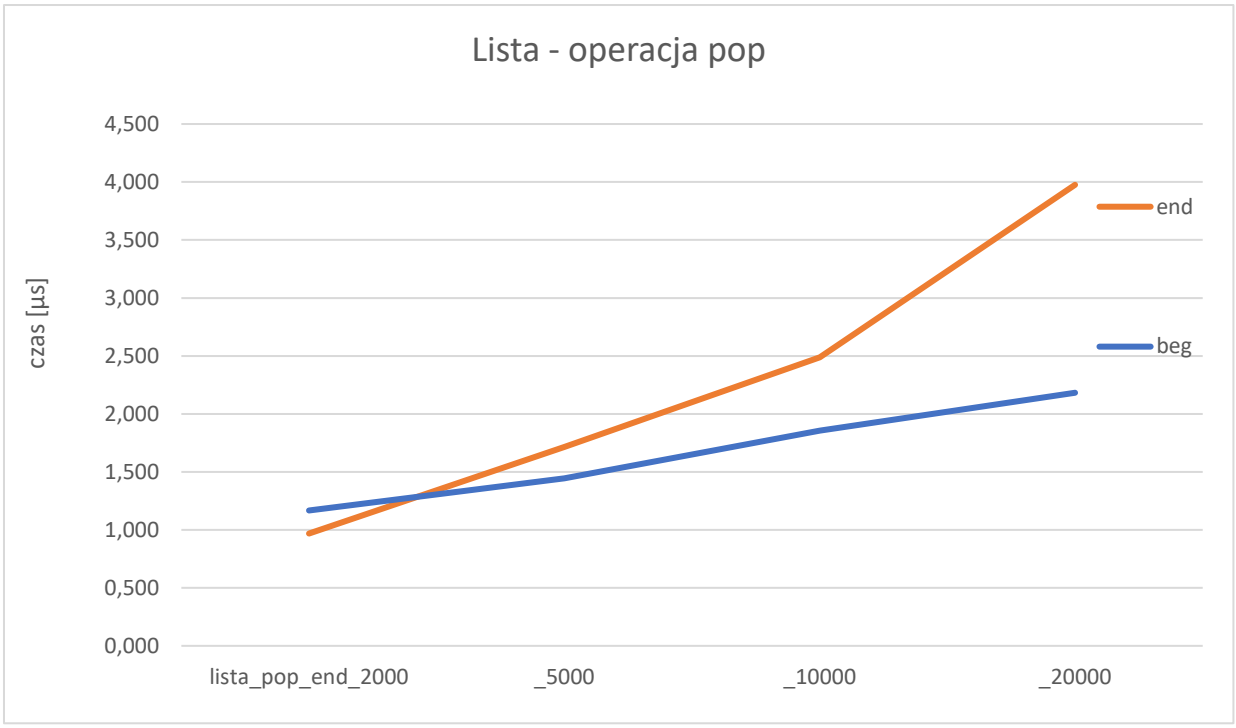

Operacja pop dla 20'000 elementów
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
80,440
77,292
58,695
Kopiec
Lista
Tablica

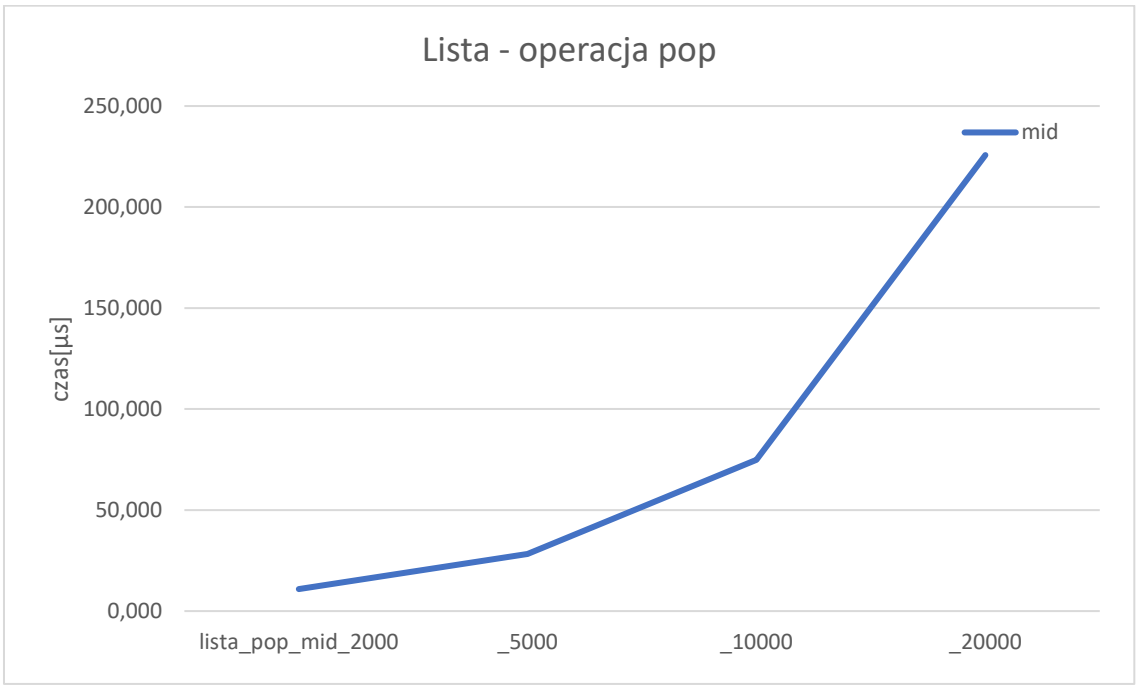

Operacja push dla 2000 elementów
12,000
10,000
8,000
6,000
4,000
2,000
0,000
czas [µs]
8,241
3,496
10,947
Kopiec
Lista
Tablica

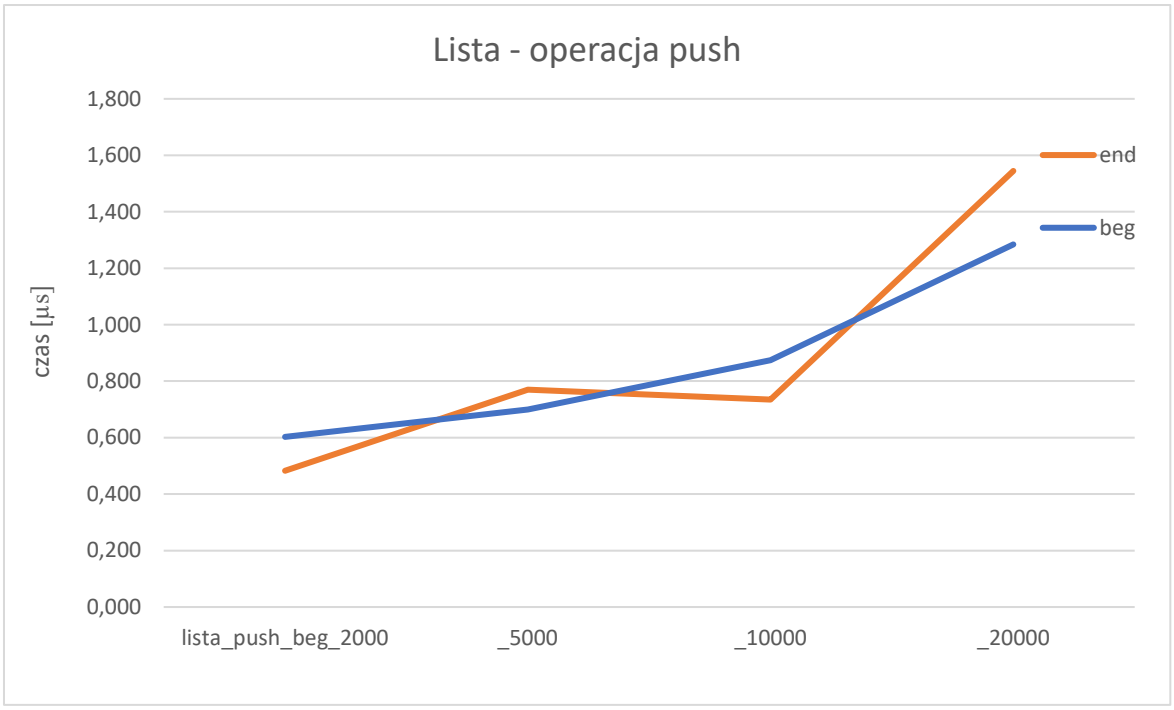

Operacja push dla 20'000 elementów
100,000
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
80,440
31,810
91,784
Kopiec
Lista
Tablica


Operacja search dla 2000 elementów
25,000
20,000
15,000
10,000
5,000
0,000
czas [µs]
21,544
7,171
5,889
Kopiec
Lista
Tablica

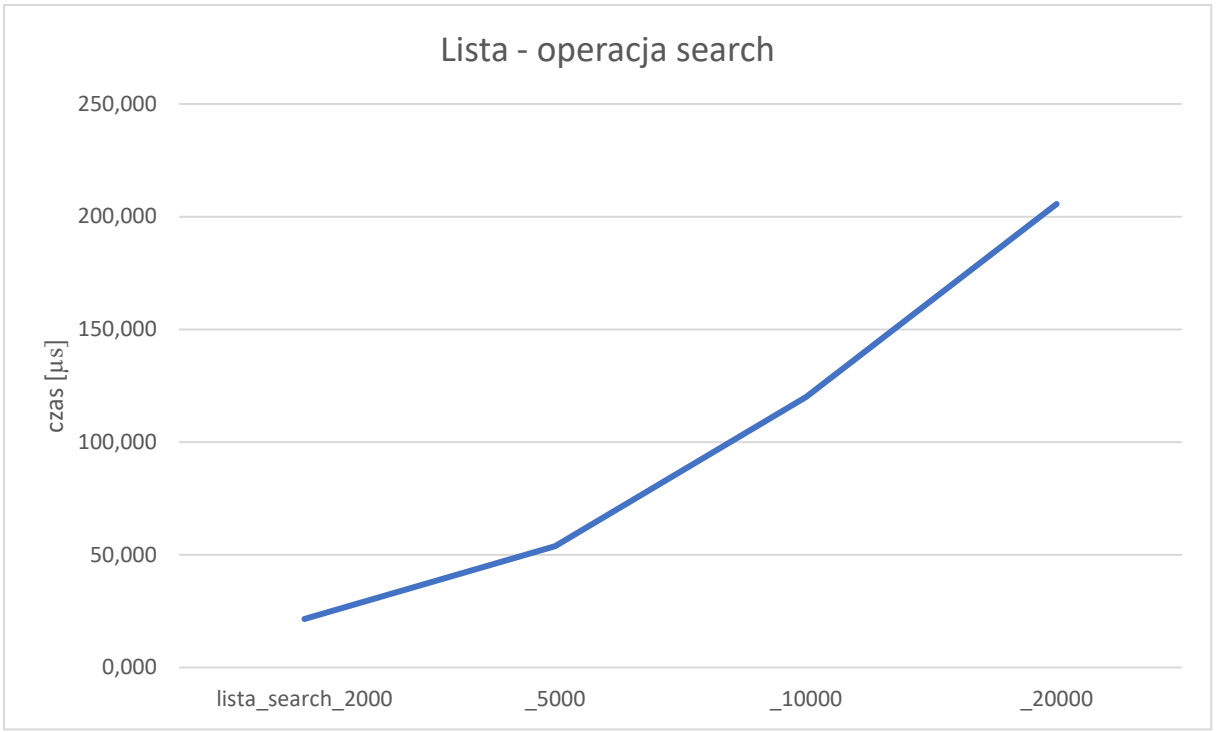

Operacja search dla 20'000 elementów
250,000
200,000
150,000
100,000
50,000
0,000
czas[µs]
205,614
53,606
46,766
Kopiec
Lista
Tablica

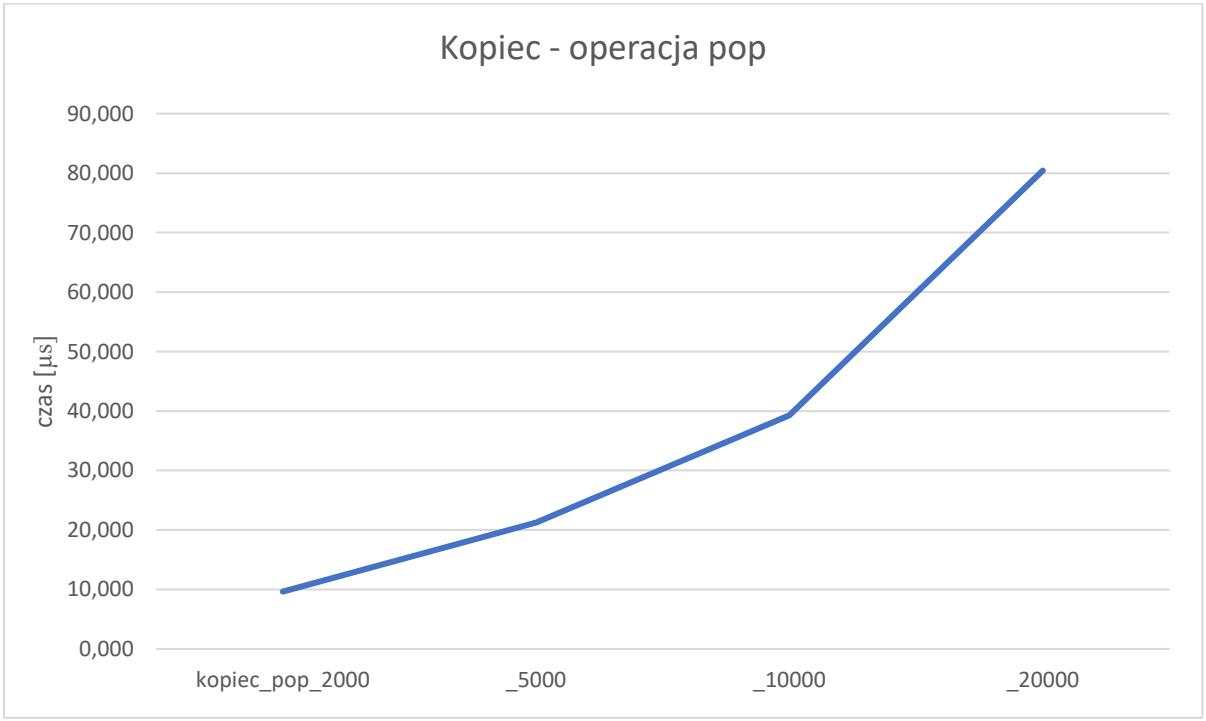

Lista - operacja pop
czas [µs]
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
0,500
0,000
lista_pop_end_2000
_5000
_10000
_20000
end
beg

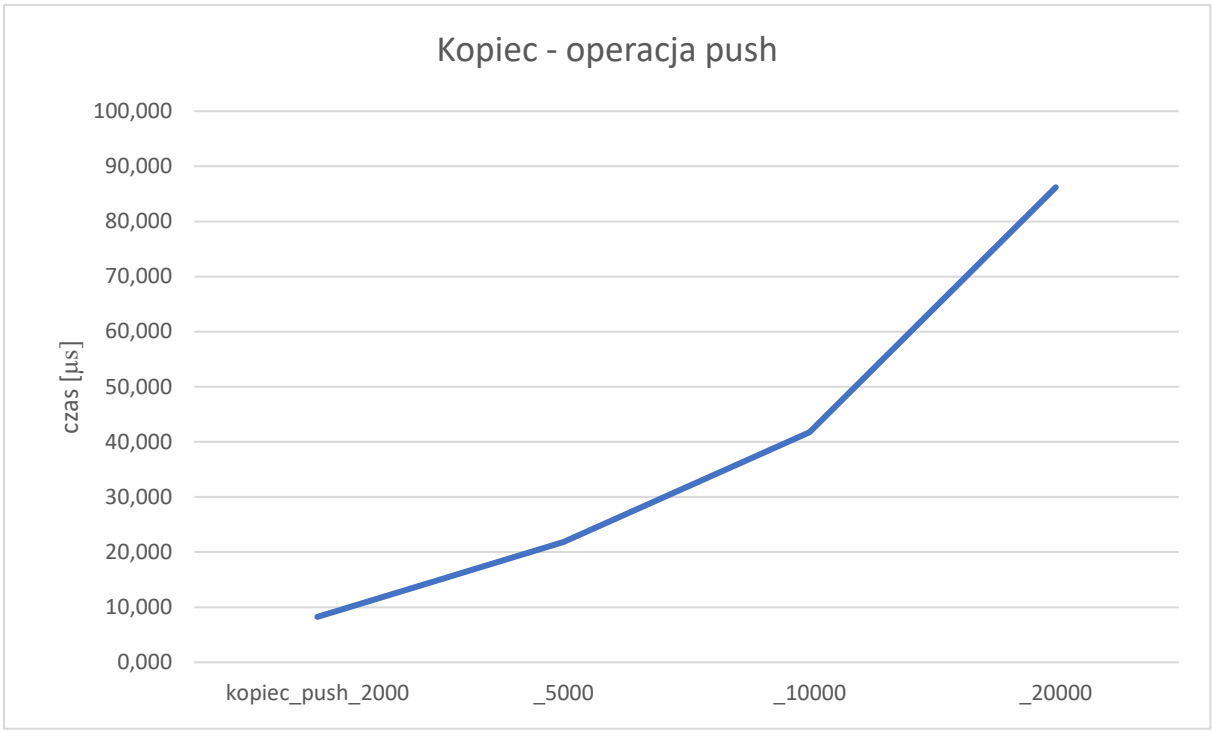

Lista - operacja pop
czas[µs]
250,000
200,000
150,000
100,000
50,000
0,000
lista_pop_mid_2000
_5000
_10000
_20000
mid

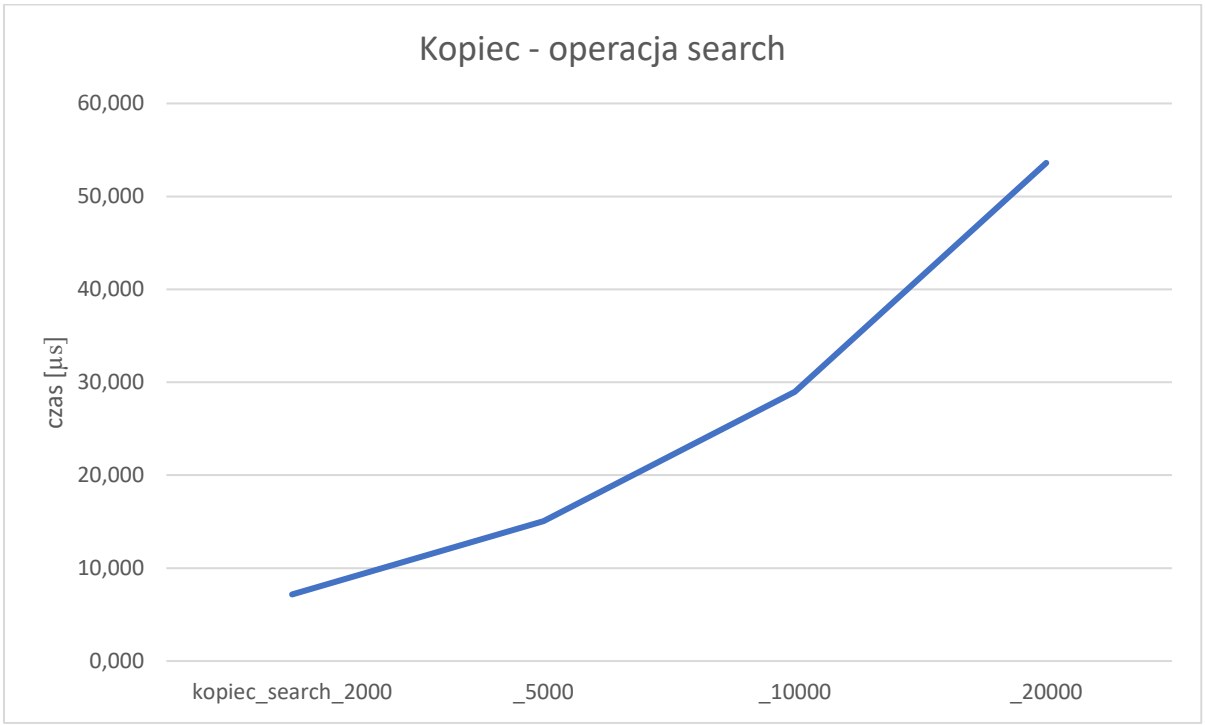

Lista - operacja push
1,800
1,600
1,400
1,200
1,000
0,800
0,600
0,400
0,200
0,000
czas [µs]
end
beg
lista_push_beg_2000
_5000
_10000
_20000

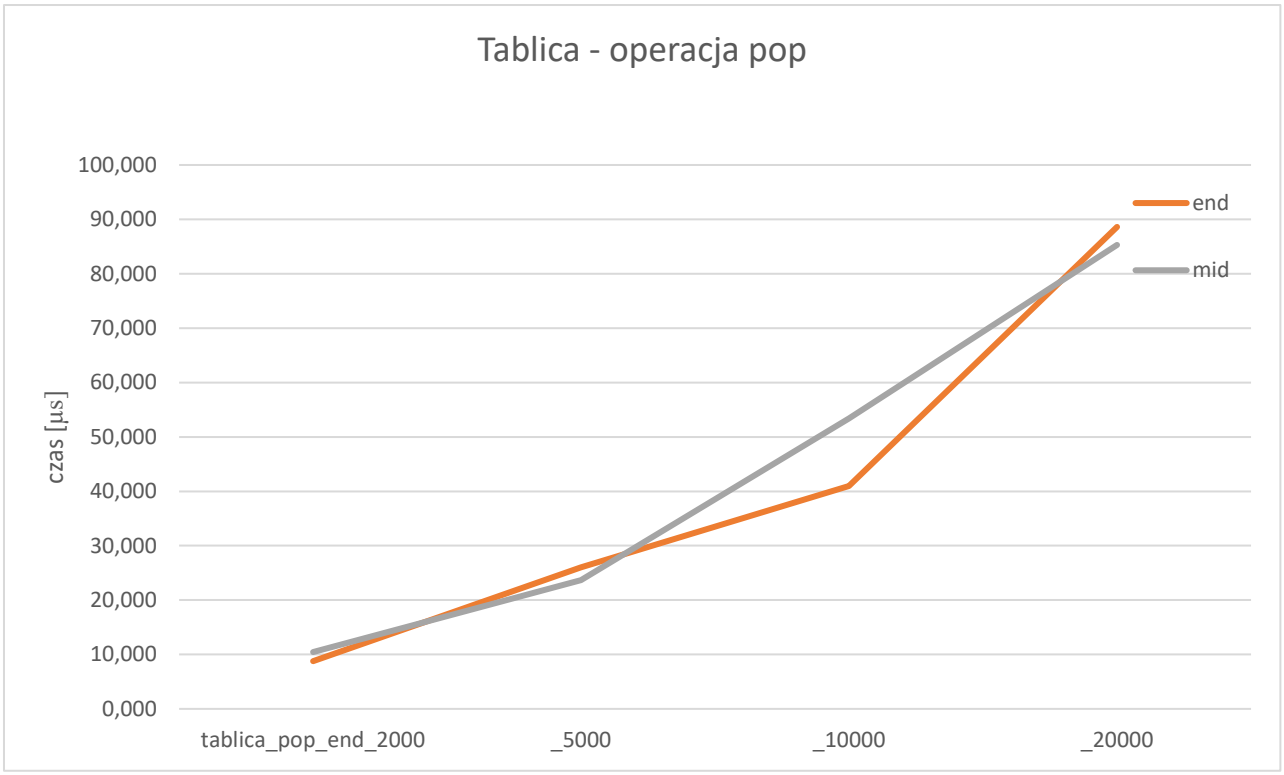

Lista - operacja push
100,000
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas[µs]
mid
lista_push_mid_2000
_5000
_10000
_20000

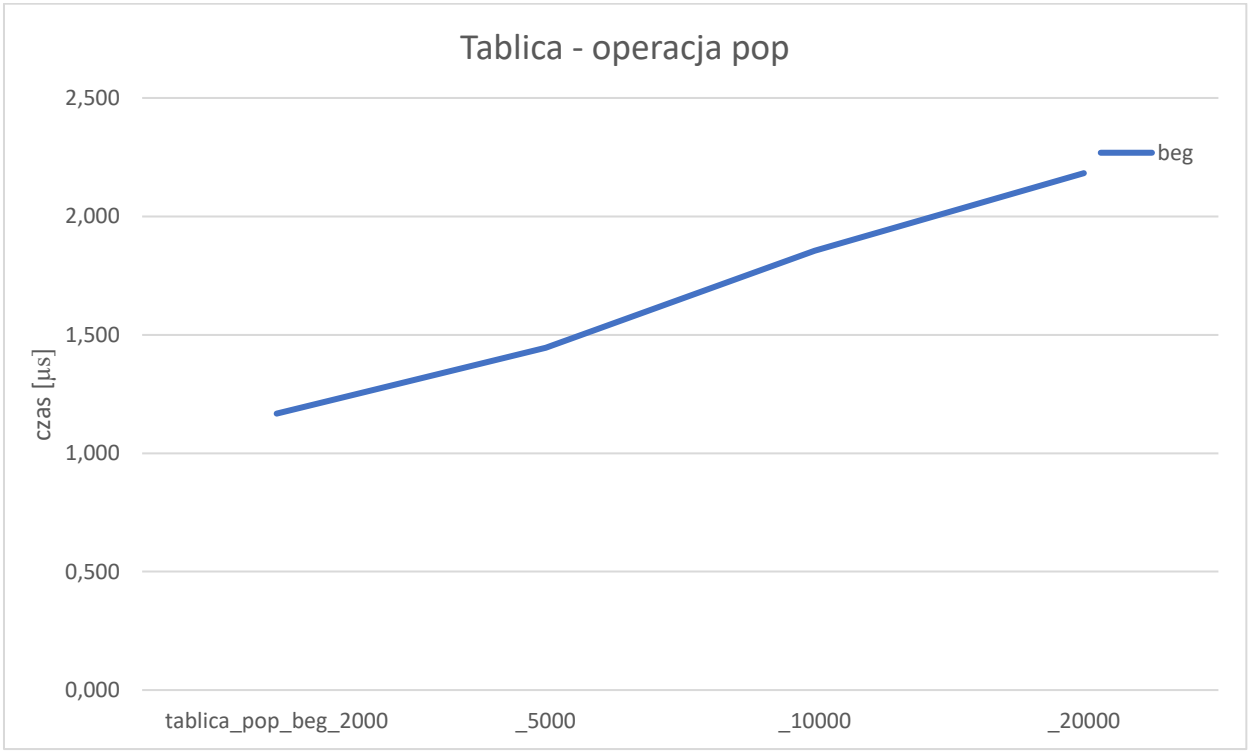

Lista - operacja search
250,000
200,000
150,000
100,000
50,000
0,000
czas [µs]
lista_search_2000
_5000
_10000
_20000

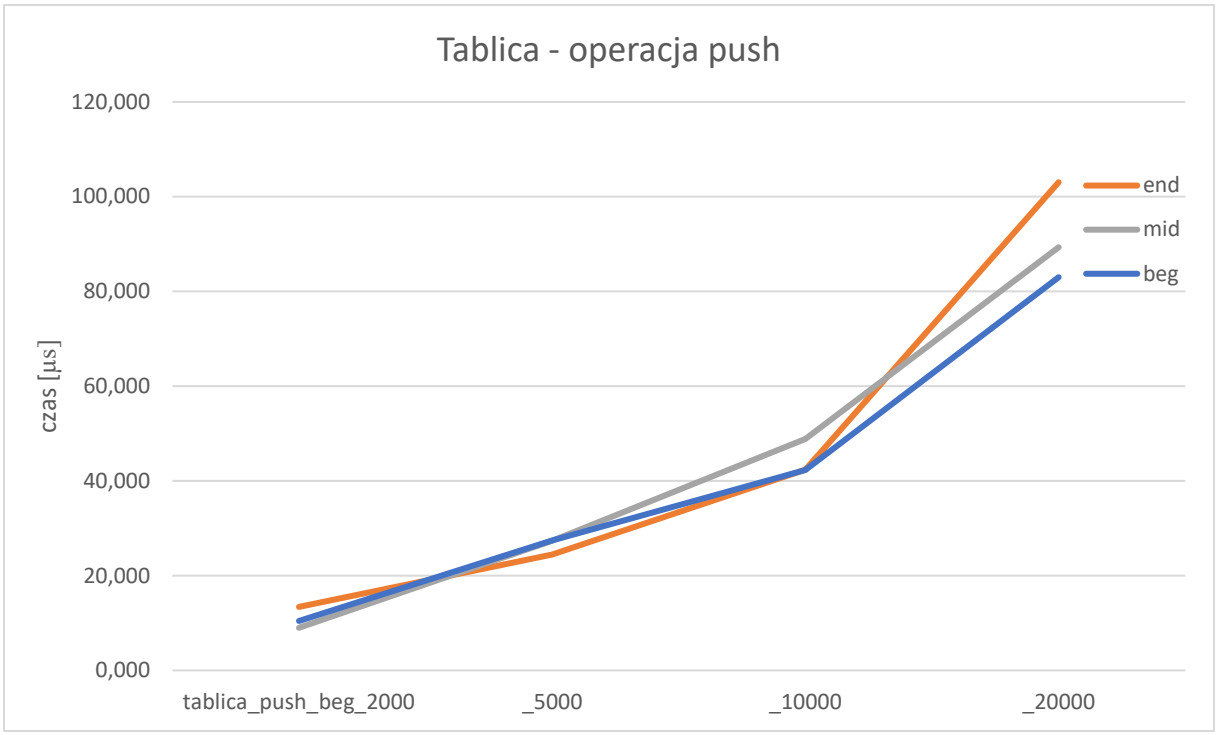

Kopiec - operacja pop
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
kopiec_pop_2000
_5000
_10000
_20000


Kopiec - operacja push
100,000
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
kopiec_push_2000
_5000
_10000
_20000


Kopiec - operacja search
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
kopiec_search_2000
_5000
_10000
_20000


Tablica - operacja pop
100,000
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0,000
czas [µs]
end
mid
tablica_pop_end_2000
_5000
_10000
_20000


Tablica - operacja pop
2,500
2,000
1,500
1,000
0,500
0,000
czas [µs]
beg
tablica_pop_beg_2000
_5000
_10000
_20000


Tablica - operacja push
120,000
100,000
80,000
60,000
40,000
20,000
0,000
czas [µs]
end
mid
beg
tablica_push_beg_2000
_5000
_10000
_20000

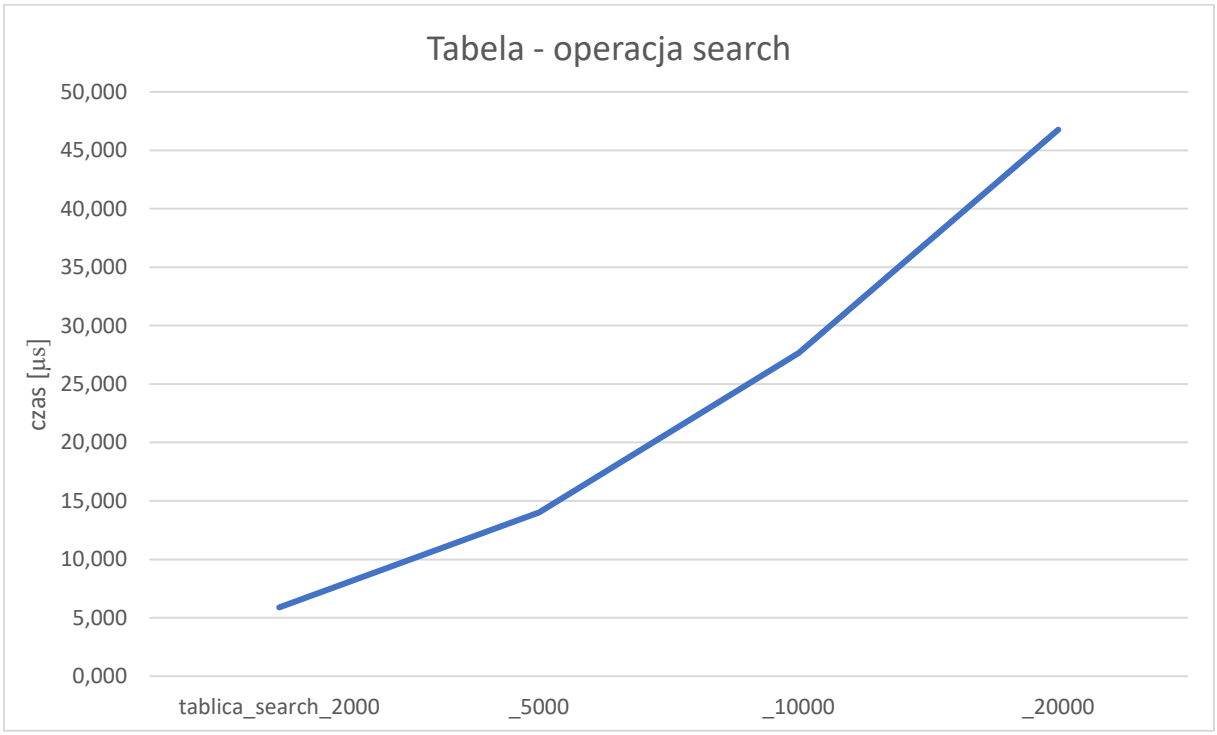


Tabela - operacja search
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0,000
czas [µs]
tablica_search_2000
_5000
_10000
_20000

6. Wnioski

Podsumowując udało mi się zaimplementować 3 działające struktury danych, każda z nich wykonuje podstawowe operacje dodania(push), usunięcia(pop) i wyszukania(search) elementu.
Tablica i lista zachowują liniową złożoność obliczeniową. Czasy operacji w tych dwóch przypadkach różnią się, ponieważ z zależności od wybranej struktury działają inne mechanizmy poruszania się po elementach i różne sposoby dostępu do danych.
Również w przypadku kopca otrzymałem złożoność liniową. Uważam, że jest to spowodowane przez wymaganą w treści dynamiczną alokację pamięci na nowej wielkości kopiec oraz związanymi z nią operacjami podstawiania.