Struktury danych	
Kierunek *Informatyczne Systemy Automatyki*	**Termin** *środa TP* $15^{15} - 16^{55}$
Imię, nazwisko, numer albumu *Piotr Brajer 272538*	**Data** *10.04.2024*
Link do Projektu https://github.com/ElemkayZ/strukturyDanych/tree/main/projekt1	

Sprawozdanie – Projekt 1

1) Średnia Złożoności obliczeniowe struktur:

a) Doubly Linked List:
   i) Wstawianie i Usuwanie na Początku lub Końcu – $O(1)$
   ii) Wstawianie i Usuwanie na Określonym Przesunięciu – $O(n)$
   iii) Wyszukiwanie Węzła z Konkretną Liczbą:
      (1) Jeśli węzeł docelowy jest bliżej początku listy, złożoność czasowa wynosi $O(n/2)$, co upraszcza się do $O(n)$
      (2) Jeśli węzeł docelowy jest bliżej końca listy, złożoność czasowa również wynosi $O(n/2)$, co upraszcza się do $O(n)$, podobnie jak w poprzednim przypadku.
   iv) Pobranie Długości Listy – $0(1)$

b) Singly Linked List:
   i) Pobieranie wskaźnika na ostatni węzeł (getLastSigNode()): $O(n)$
   ii) Wstawianie nowego węzła na końcu (addEnd()): $O(n)$
   iii) Wstawianie nowego węzła na początku (addStart()): $O(1)$
   iv) Usuwanie pierwszego węzła (removeStart()): O(1)
   v) Usuwanie węzła na podanym przesunięciu (remove()): $O(n)$
   vi) Pobieranie długości listy (getLen()): $O(n)$
   vii) Wstawianie węzła na podanym przesunięciu (add()): $O(n)$
   viii) Usuwanie ostatniego węzła (removeEnd()): $O(n)$
   ix) Wyszukiwanie węzła z konkretną liczbą (FindNumber()): $O(n)$

c) Better Linked List:
   i) Dodawanie nowego węzła na końcu (addEnd()): $O(1)$
   ii) Dodawanie nowego węzła na początku (addStart()): $O(1)$
   iii) Usuwanie pierwszego węzła (removeStart()): $O(1)$
   iv) Pobieranie wskaźnika na głowę (getHead()): $O(1)$
   v) Pobieranie wskaźnika na ogon (getTail()): $O(1)$
   vi) Usuwanie węzła na podanym przesunięciu (remove()): $O(n)$
   vii) Pobieranie długości listy (getLen()): $O(n)$
   viii) Dodawanie węzła na podanym przesunięciu (add()): $O(n)$
   ix) Usuwanie ostatniego węzła (removeEnd()): $O(n)$
   x) Wyszukiwanie węzła z daną liczbą (FindNumber()): $O(n)$

d) ArrayList:
   i) Dodawanie, Wstawianie, Pobieranie i Usuwanie Elementów: $O(1)$
   ii) Wyszukiwanie i Wypisywanie: $O(n)$

2) Założenia projektowe:

a) Wielkość struktur:

i) Każda struktura została badana na: 5 000, 10 000, 15 000, 20 000, 40 000, 60 000, 80 000, 100 000 ilości elementów

b) Sposób generowania elementów do struktur:

i) Każda struktura miała wczytane wartości z pliku RandomNumber.txt w którym zostały wygenerowane przypadkowe liczby

```
srand(time(0));
for (int i = 0; i < numNumbers; ++i) {
    int randomNumber = rand();
    outFile << randomNumber << endl;
}
outFile.close();
```

c) Sposób Pomiaru czasu funkcji struktury:

i) Przykład sposobu pomiaru czasu funkcji:

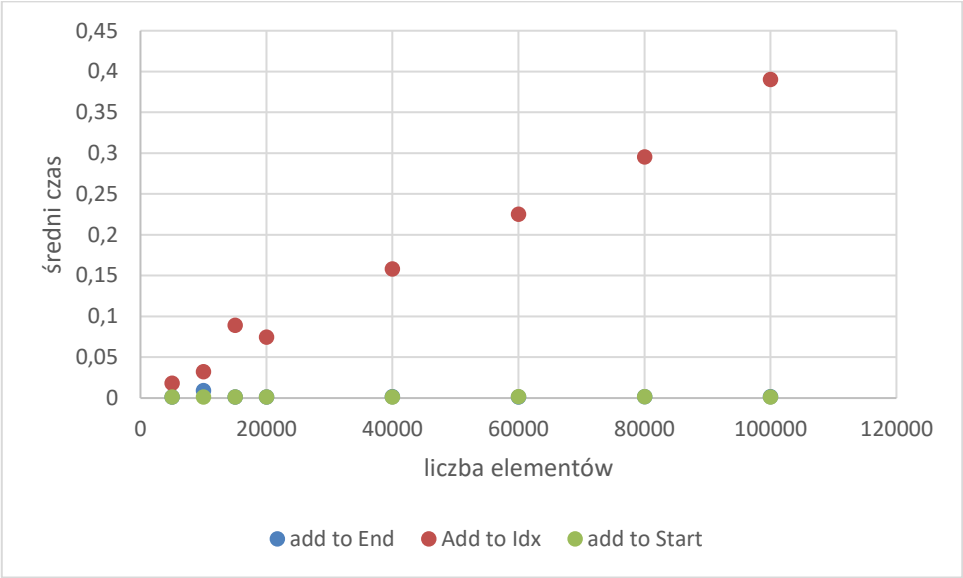
```
auto start = std::chrono::high_resolution_clock::now();
dubLinklist->addStart(_data);
auto stop = std::chrono::high_resolution_clock::now();
d = stop - start;
std::cout << "Time taken = " << d.count() << std::endl;
```

3) Badania:

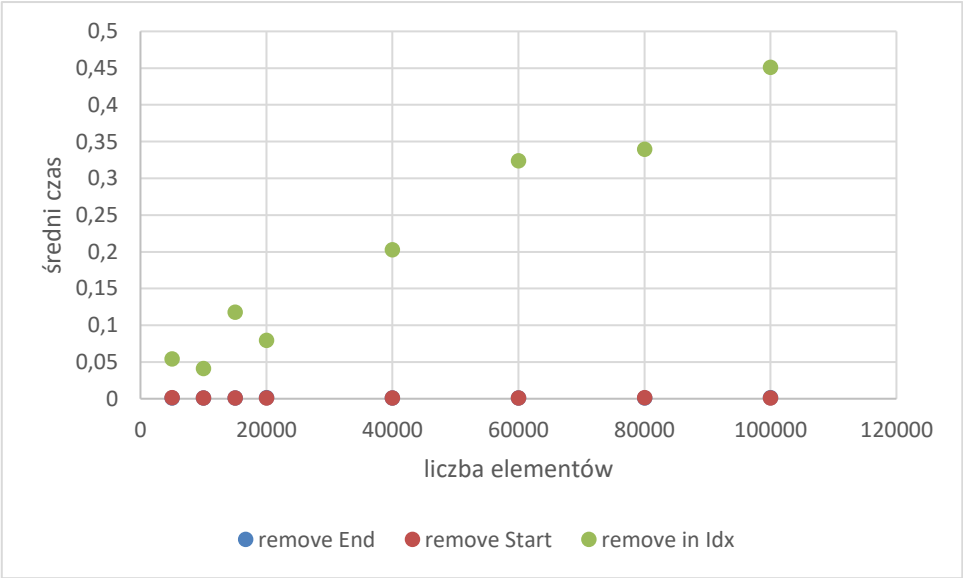
a) Doubly Linked List:

Tabela 1: średnie pomiary czasu wykonania funkcji

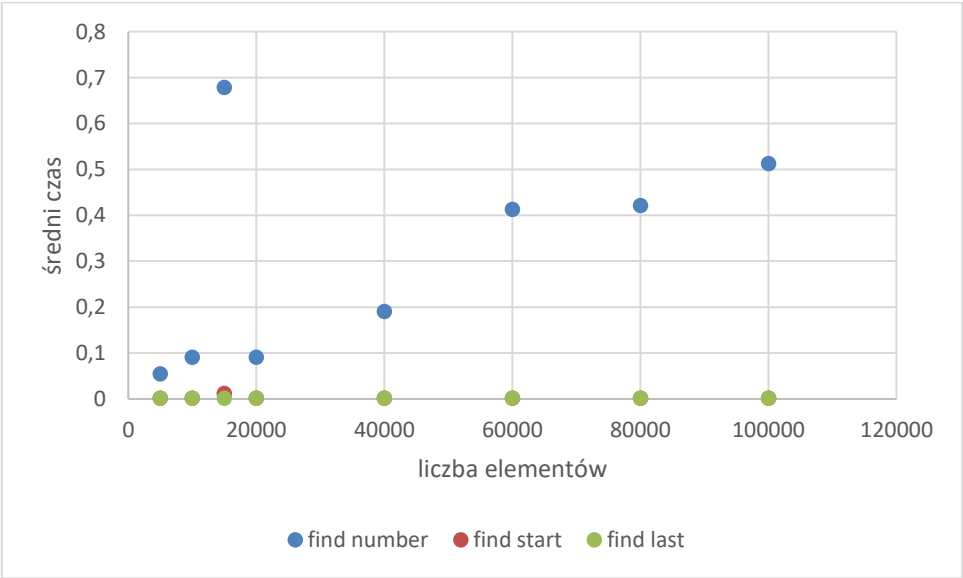
	czasy [ms]							
funkcje	5000	10000	15000	20000	40000	60000	80000	100000
add to End	0,0011	0,009	0,0013	0,0012	0,0014	0,0011	0,0014	0,0014
Add to Idx	0,018	0,0322	0,089	0,0744	0,1579	0,225	0,2952	0,3903
add to Start	0,0013	0,0013	0,0013	0,0013	0,0013	0,0014	0,0015	0,0012
remove End	0,001	0,0011	0,0011	0,0014	0,0011	0,0011	0,001	0,0016
remove Start	0,0016	0,001	0,0013	0,001	0,0012	0,0011	0,0015	0,001
remove in Idx	0,05429	0,0414	0,1178	0,0794	0,2027	0,3238	0,3397	0,4512
find number	0,0543	0,0908	0,678	0,0903	0,1904	0,4124	0,4211	0,512
find start	0,001	0,001	0,012	0,001	0,0011	0,0011	0,0012	0,0012
find last	0,001	0,0011	0,0011	0,0012	0,0012	0,0012	0,0013	0,0011



WYKRES 1: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA DODAWANIA DLA DOUBLY LINKED LIST



WYKRES 2: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA USÓWANIA DLA DOUBLY LINKED LIST

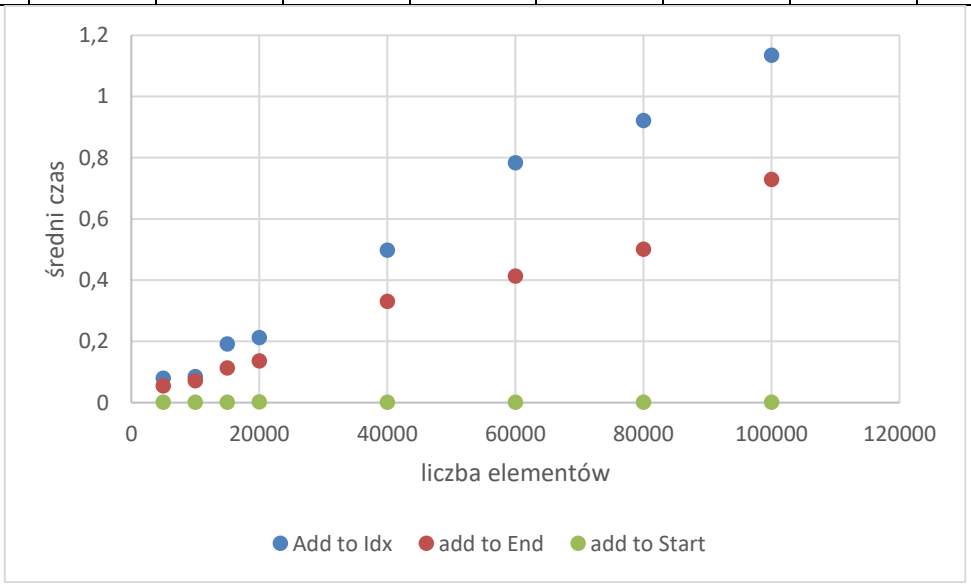


WYKRES 3: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA WYSZUKIWANIA DLA DOUBLY LINKED LIST

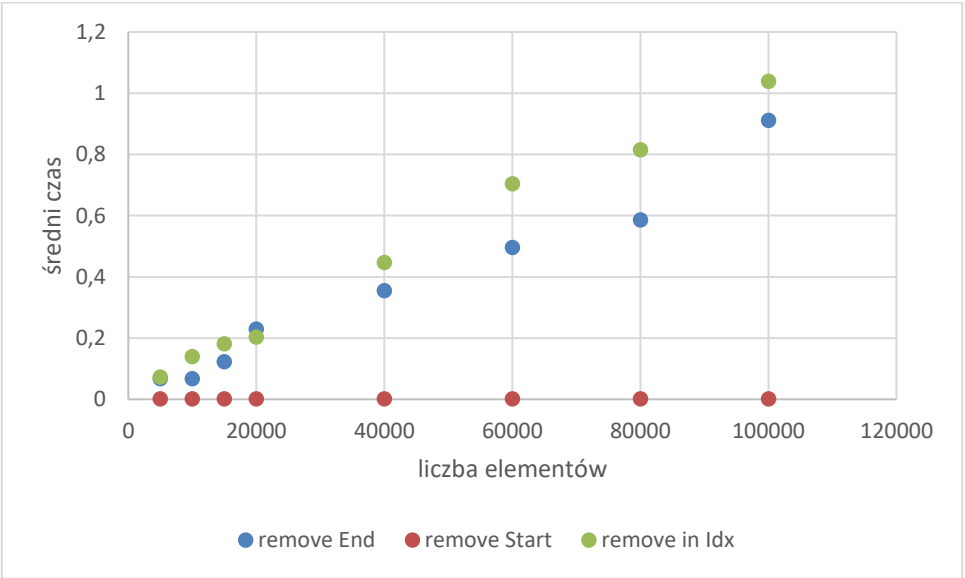
b) Singly Linked List:

Tabela 2: średnie pomiary czasu wykonania funkcji

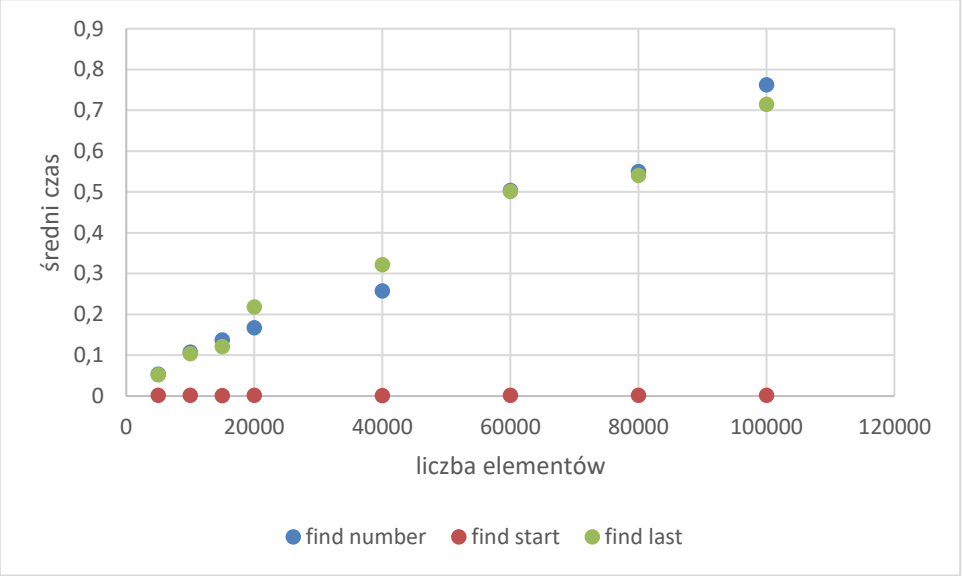
	czasy [ms]							
funkcje	5000	10000	15000	20000	40000	60000	80000	100000
add to End	0,0551	0,0716	0,1139	0,1362	0,3311	0,4139	0,5017	0,7295
Add to Idx	0,0801	0,0855	0,1924	0,2129	0,4978	0,7841	0,9214	1,1352
add to Start	0,0012	0,0014	0,0012	0,0022	0,0013	0,0012	0,0013	0,0012
remove End	0,067	0,067	0,1223	0,2289	0,355	0,4954	0,5858	0,9111
remove Start	0,0013	0,0013	0,0013	0,0013	0,0014	0,0013	0,0014	0,0012
remove in Idx	0,0721	0,1391	0,1809	0,2027	0,4464	0,7041	0,8145	1,0381
find number	0,0531	0,1071	0,1374	0,1671	0,2569	0,5038	0,5501	0,7625
find start	0,0012	0,0012	0,0011	0,0012	0,0011	0,0012	0,0012	0,0012
find last	0,052	0,1036	0,121	0,2184	0,3214	0,5012	0,54	0,7142



WYKRES 4: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA DODAWANIA DLA SINGLY LINKED LIST



WYKRES 5: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA USÓWANIA DLA SINGLY LINKED LIST

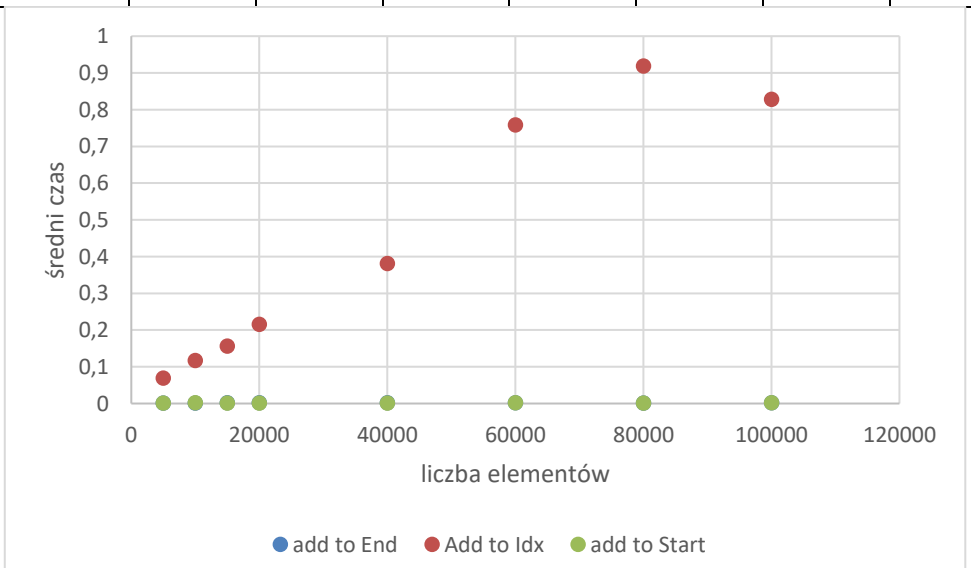


WYKRES 6: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA WYSZUKIWANIA DLA SINGLY LINKED LIST

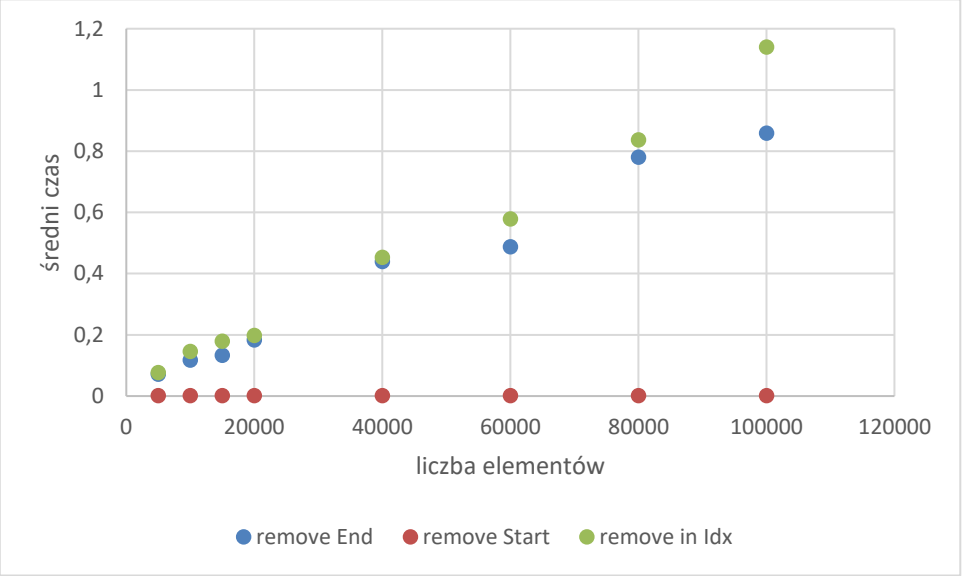
c) Better Linked List:

Tabela 3: średnie pomiary czasu wykonania funkcji

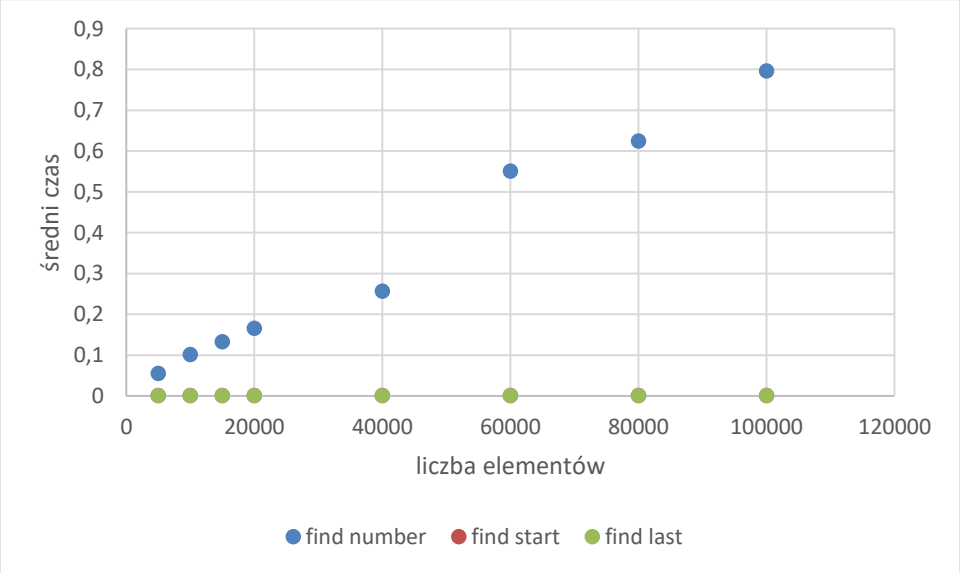
	czasy [ms]							
funkcje	5000	10000	15000	20000	40000	60000	80000	100000
add to End	0,0012	0,0012	0,0015	0,0016	0,0014	0,0017	0,0011	0,0013
Add to Idx	0,0687	0,1166	0,156	0,2155	0,3811	0,7581	0,9184	0,8279
add to Start	0,0012	0,0015	0,0012	0,0011	0,0011	0,0014	0,0011	0,0014
remove End	0,071	0,1169	0,1328	0,1828	0,4396	0,487	0,78	0,8587
remove Start	0,0012	0,0012	0,0012	0,0012	0,0011	0,0012	0,0011	0,001
remove in Idx	0,0763	0,1454	0,1783	0,1981	0,4525	0,5782	0,8363	1,1399
find number	0,0548	0,1015	0,1325	0,1657	0,2564	0,5511	0,6241	0,7959
find start	0,0012	0,0012	0,0012	0,0012	0,0011	0,0013	0,0012	0,0011
find last	0,0012	0,0012	0,0012	0,0012	0,0013	0,0012	0,0013	0,0012



WYKRES 7: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA DODAWANIA DLA SINGLY LINKED LIST



WYKRES 8: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA USÓWANIA DLA SINGLY LINKED LIST

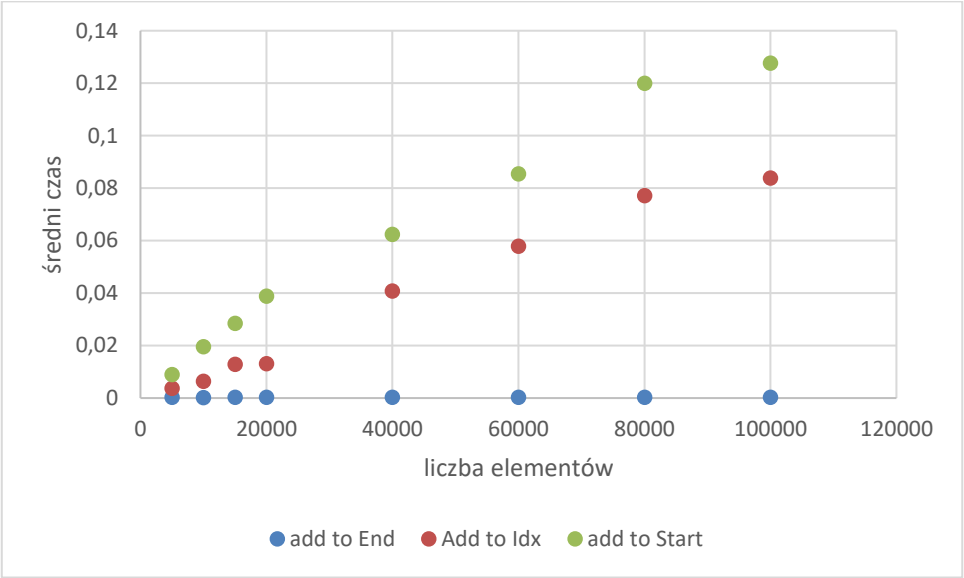


WYKRES 9: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA WYSZUKIWANIA DLA SINGLY LINKED LIST

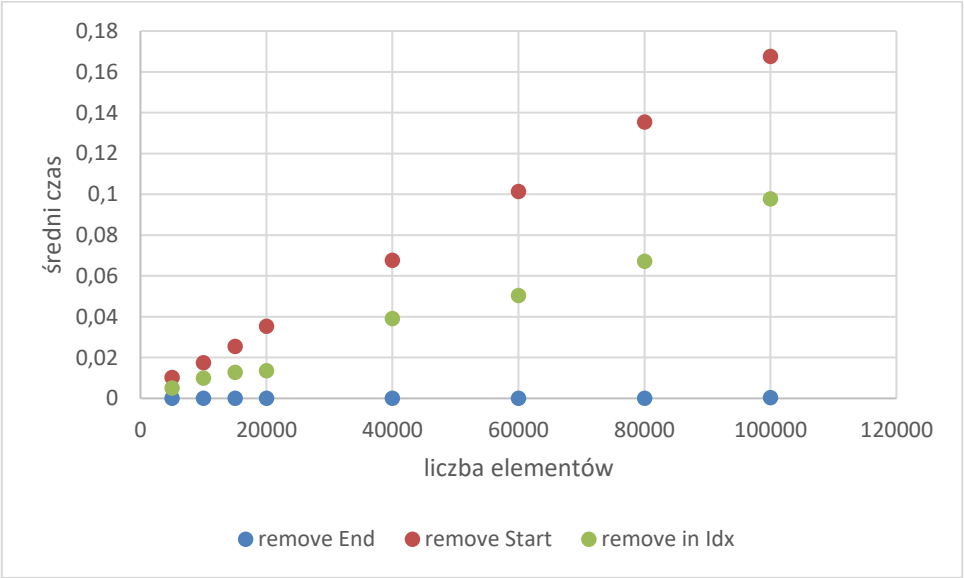
d) ArrayList:

Tabela 4: średnie pomiary czasu wykonania funkcji

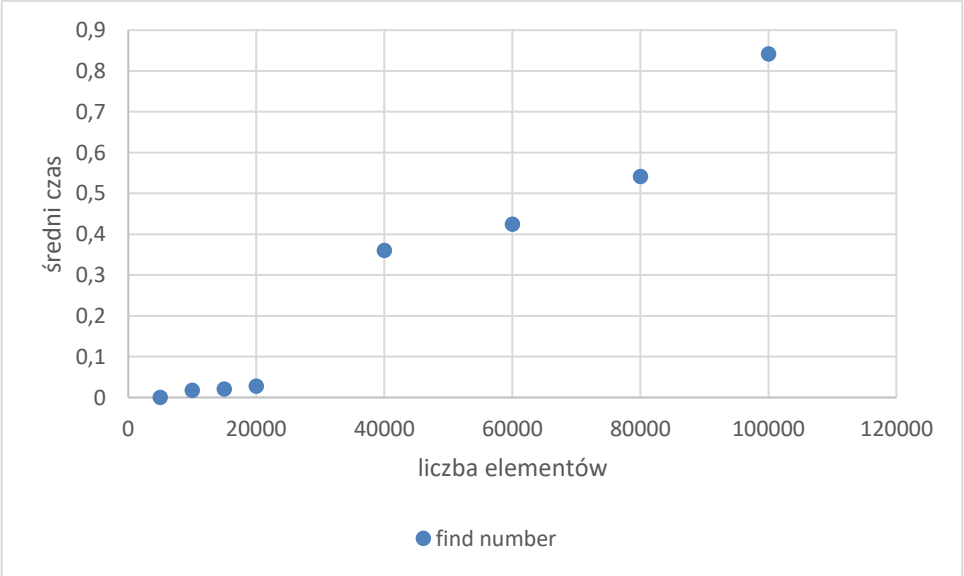
	czasy [ms]							
funkcje	5000	10000	15000	20000	40000	60000	80000	100000
add to End	0,0002	0,0001	0,0002	0,0002	0,0003	0,0002	0,0003	0,0002
Add to Idx	0,0037	0,0064	0,0128	0,013	0,0408	0,0578	0,0771	0,0838
add to Start	0,0089	0,0195	0,0284	0,0388	0,0623	0,0854	0,1199	0,1276
remove End	0,0002	0,0001	0,0002	0,0002	0,0001	0,0002	0,0001	0,0005
remove Start	0,0103	0,0175	0,0255	0,0354	0,0677	0,1014	0,1355	0,1676
remove in Idx	0,0052	0,01	0,0128	0,0136	0,0392	0,0505	0,0672	0,0978
find number	0,0003	0,0175	0,0206	0,0275	0,36	0,4241	0,5412	0,8412



WYKRES 10: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA DODAWANIA DLA ARRAYLIST



WYKRES 11: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA USÓWANIA DLA ARRAYLIST



WYKRES 12: PORÓWNANIE ŚREDNIEGO CZASU WYKONANIA WYSZUKIWANIA DLA ARRAYLIST

4) Wnioski:

Względem siebie wszystkie struktury posiadają odpowiedni czas wykonywania. Można zauważyć że czasami pomiary posiadają mocniejsze zróżnicowanie wyniki czasowe w porównaniu do teoretycznie możliwych czasów co może być spowodowane nie idealnym środowiskiem do zbierania pomiarów(zbyt mocno używany komputer).
Można zauważyć że jedno kierunkowe listy pozwalają na efektywniejsze modyfikowanie początkowych wartości, natomiast arrayList jest lepszy w stosunku do końcowych wartości, podczas gdy lista dwukierunkowa jest czasowo pomiędzy ArrayList a listami jednokierunkowymi.