Sprawozdanie ze struktur danych i złożoności obliczeniowej, projekt 1

Jakub Klawon

Grupa poniedziałek godzina 15 TP

Spis treści

1 Wstęp

Poniższe sprawozdanie dotyczy projektu ze struktur danych i złożoności obliczeniowej. Przedstawia efekty obiektowej implementacji tablicy dynamicznej, listy dwukierunkowej, kopca oraz drzewa czerwono-czarnego. Na każdej strukturze wykonane zostały pomiary czasu wykonywania ich funkcji. Wyniki pomiarów czasów podane są w nanosekundach (10^{-9}). Każda operacja została przywołana 300 razy za każdym razem tworząc nowy zestaw losowych liczb w wielkości zestawów 1000, 2000, 5000, 10000, 20000, 30000 oraz 50000. Dla każdej wielkości przedstawiono 15 wyników, z których obliczona została średnia arytmetyczna. Wykresy stworzone z wartości na osi y wskazują na czas wykonania [w sekundach] a oś x pokazuje wielkość zestawu.

2 Tablica dynamiczna

2.1 Opis

Tablica dynamiczna tworzona jest w momencie uruchomienia programu. W każdym momencie można usunąć ją z pamięci, dzięki czemu efektywniej wykorzystuje zasoby pamięciowe. Funkcje zaimplementowane do tej struktury to: dodawanie na początek, koniec oraz w dowolnym miejscu (w tym przypadku jest to część bliższa końca, ale nie sam koniec, gdyż program analizuje podany indeks i przywołuje funkcję "na początek"i "na koniec"), usuwanie z początku, końca i w dowolnym miejscu oraz wyszukiwanie liczby.

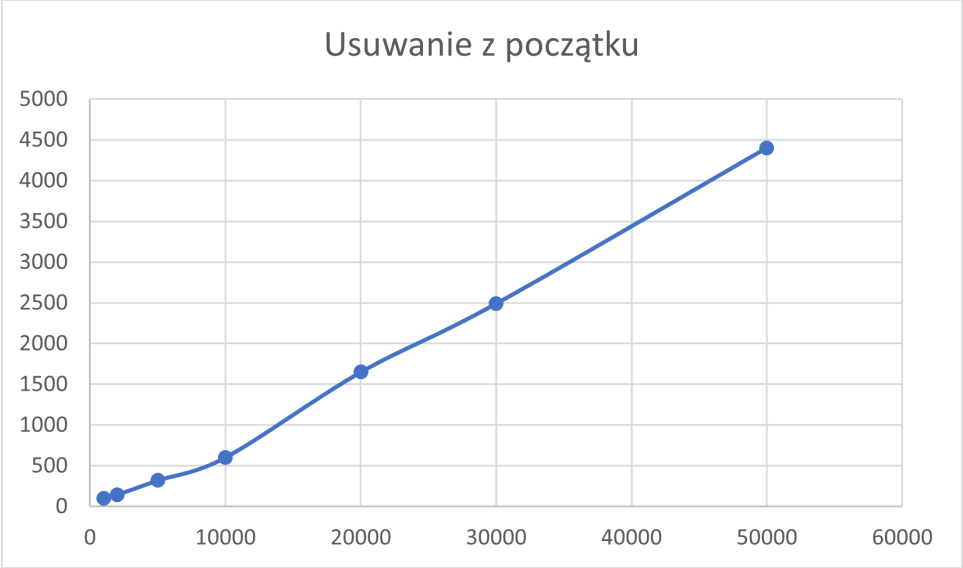
2.2 Pomiary

Ilość elementów	Dodawanie koniec	Usuwanie koniec	Dodawanie początek	Usuwanie początek
1000	149,67	168,20	143,72	102,83
2000	148,04	255,41	187,74	145,30
5000	335,06	454,70	426,97	321,27
10000	592,03	615,71	838,39	602,81
20000	1513,69	1554,65	2022,096	1650,95
30000	2440,93	2470,29	2954,86	2491,64
50000	4314,61	4379,87	4504,27	4402,94

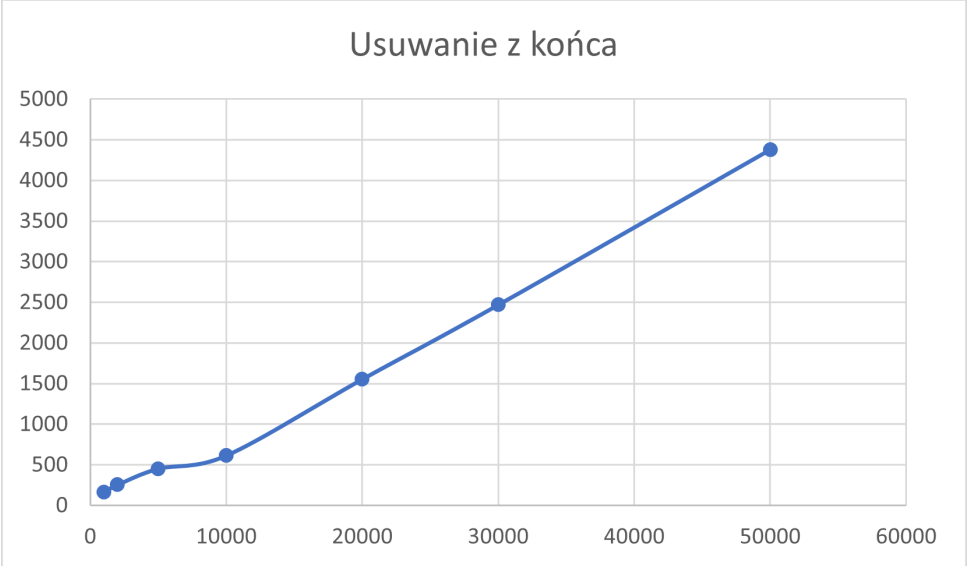
Ilość elementów	Dodawanie indeks	Usuwanie indeks	Wyszukiwanie
1000	114,76	109,27	205,93
2000	175,81	375,20	186,42
5000	470,46	424,40	880,78
10000	862,56	740,40	1696,29
20000	2002,63	1777,69	3354,12
30000	2949,51	2719,87	4873,70
50000	4593,28	4555,10	8059,64

2.3 Wykresy

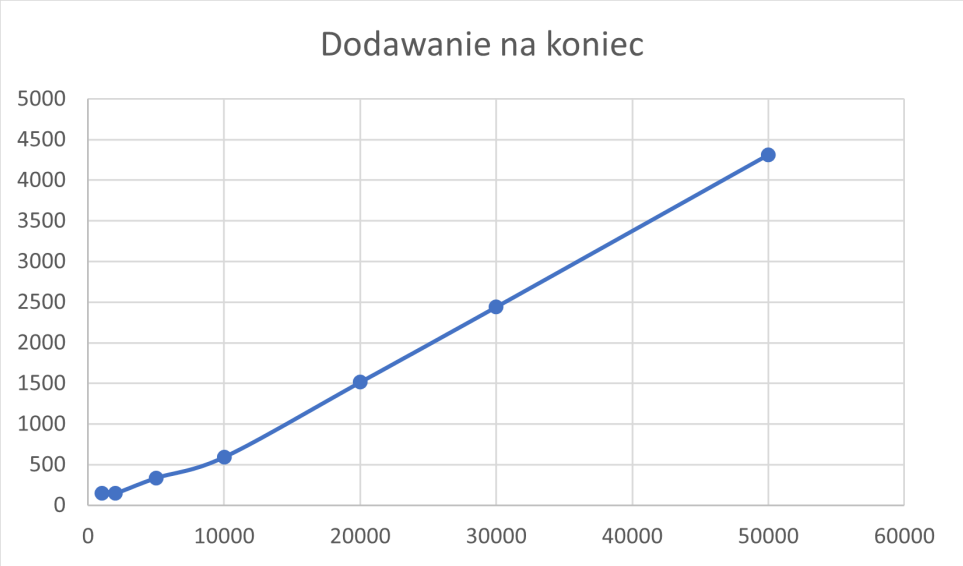
Rysunek 1: Dodawanie na początek w liście



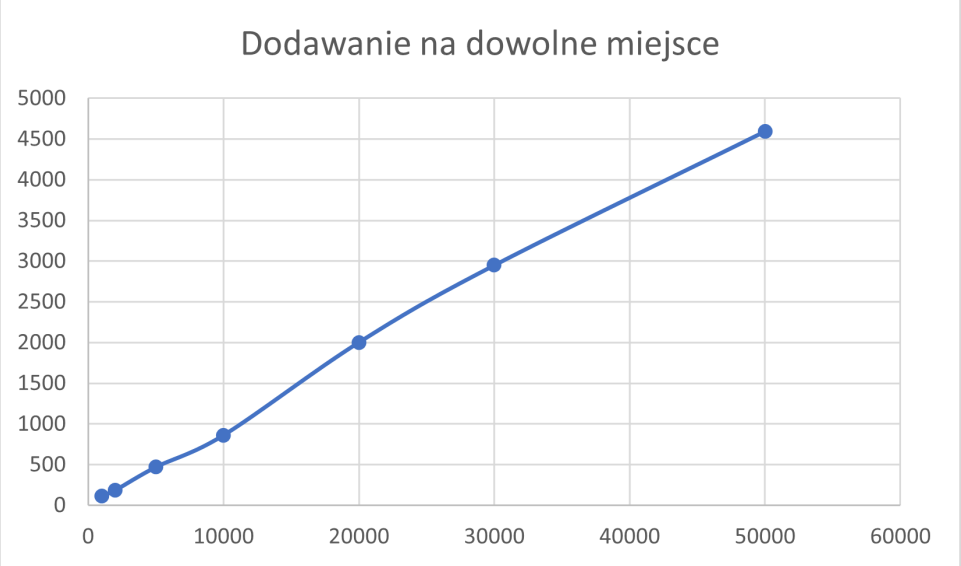
Rysunek 2: Usuwanie z początku w liście



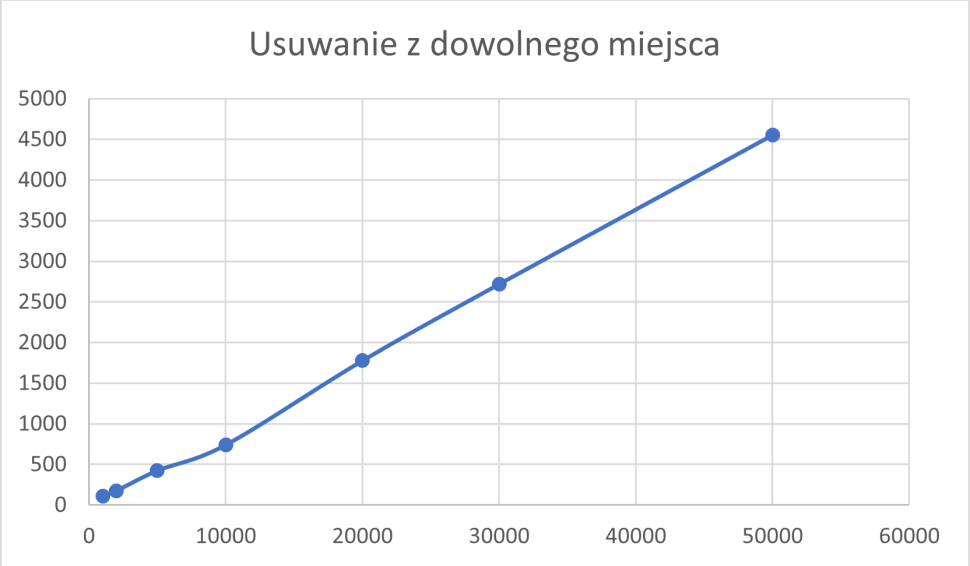
Rysunek 3: Usuwanie z końca w liście



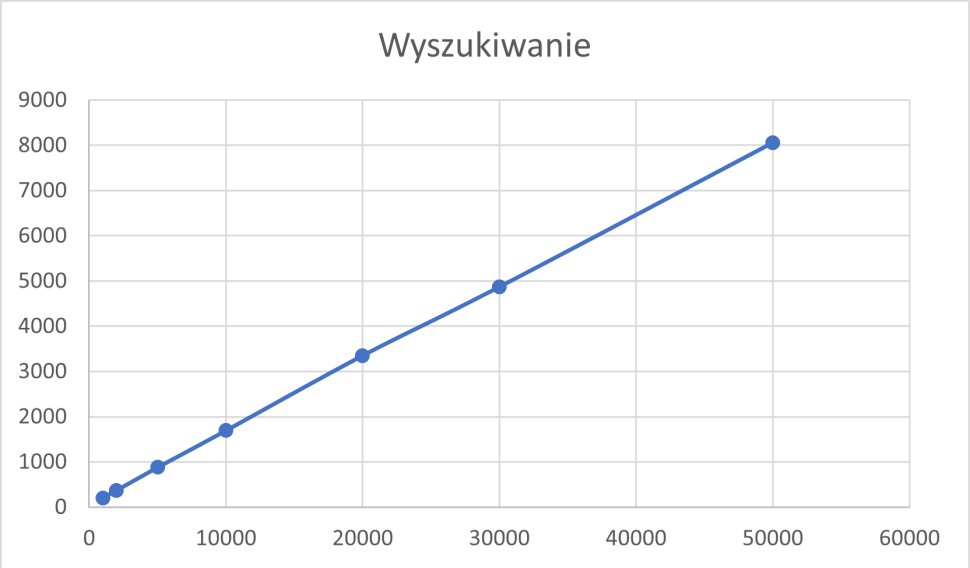
Rysunek 4: Dodawanie na końcu w liście



Rysunek 5: Dodawanie w dowolne miejsce w liście



Rysunek 6: Usuwanie z dowolnego miejsca w liście



Rysunek 7: Wyszukiwanie w liście

2.4 Wnioski

Analizując wyniki pomiarów można zauważyć, że we wszystkich funkcjach złożoność obliczeniowa wynosi O(n). Operacje na tablicy dynamicznej zajmują całkiem dużo czasu w porównaniu do innych struktur. W przypadku wyszukiwania jest znacznie wydajniejsza niż lista dwukierunkowa.

3 Lista dwukierunkowa

3.1 Opis

Lista dwukierunkowa jest zbiorem elementów o trzech atrybutach: dana liczbowa, wskaźnik na poprzednika i wskaźnik na następnika. Dzięki temu niektóre operacje wykonuje się znacznie szybciej. W przypadku, kiedy potrzebujemy dostać się do danego indeksu, możemy zacząć od końca albo od

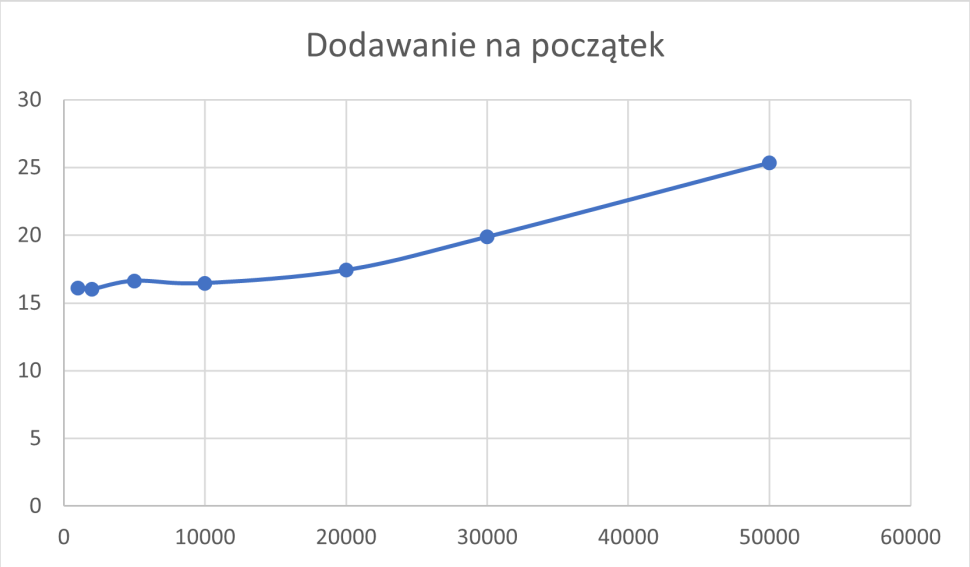
początku co jest dużym plusem w porównaniu do listy jednokierunkowej. Lista pochłania jednak duże zasoby pamięciowe w porównaniu do tablicy.

3.2 Pomiary

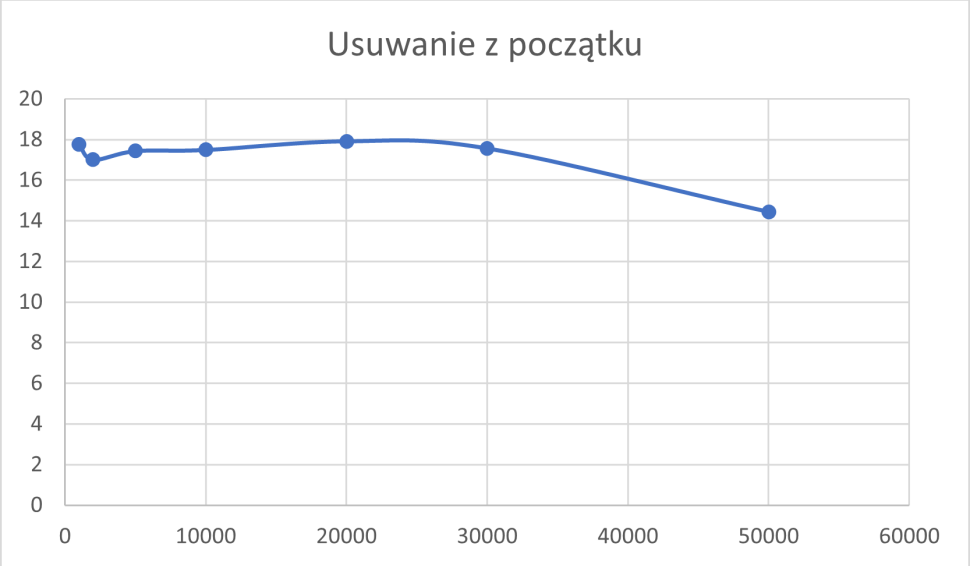
Ilość elementów	Dodawanie koniec	Usuwanie koniec	Dodawanie początek	Usuwanie początek
1000	16,51	17,09	16,11	17,75
2000	16,24	16,54	16,02	17,01
5000	16,73	16,94	16,63	17,43
10000	16,57	17,36	16,45	17,48
20000	17,36	18,70	17,41	17,91
30000	15,73	17,22	19,87	17,56
50000	14,75	16,15	25,34	14,44

Ilość elementów	Dodawanie indeks	Usuwanie indeks	Wyszukiwanie
1000	510,92	466,38	493,63
2000	1229,64	947,39	1007,81
5000	3606,33	2400,89	2564,76
10000	6946,97	6451,82	6566,15
20000	14295,33	13520,10	13692,81
30000	20127,58	19217,8	19709,34
50000	34026,41	32052,16	32250,35

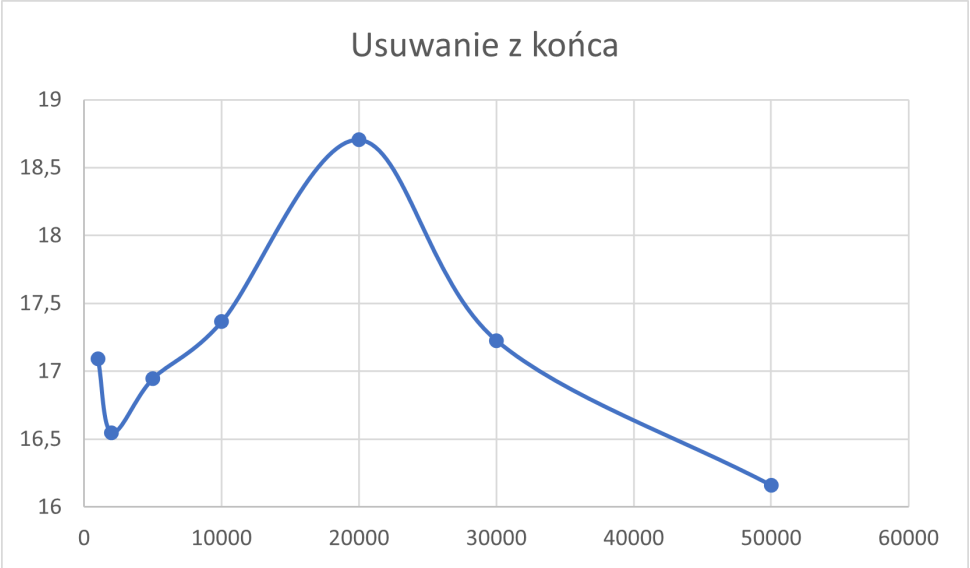
3.3 Wykresy



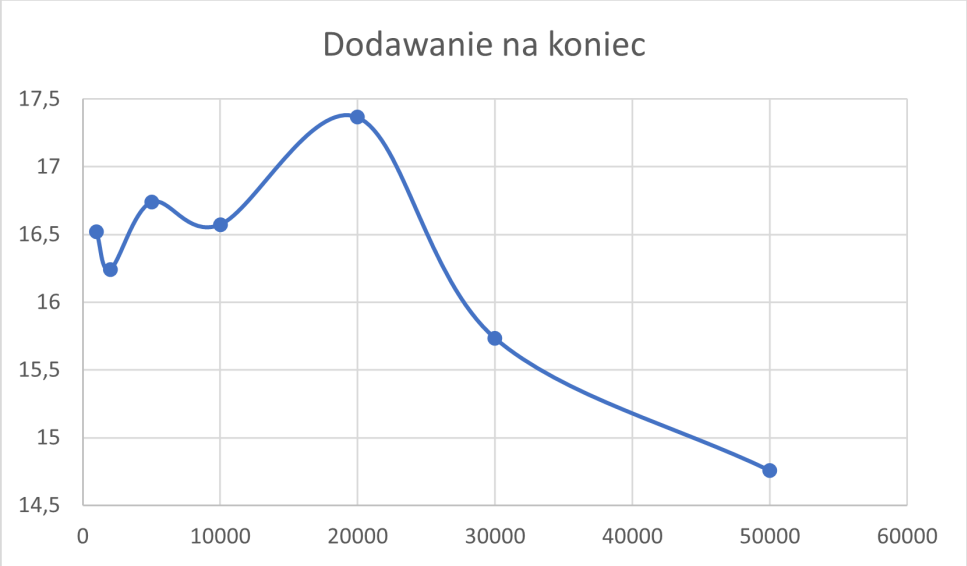
Rysunek 8: Dodawanie na początek w liście



Rysunek 9: Usuwanie z początku w liście

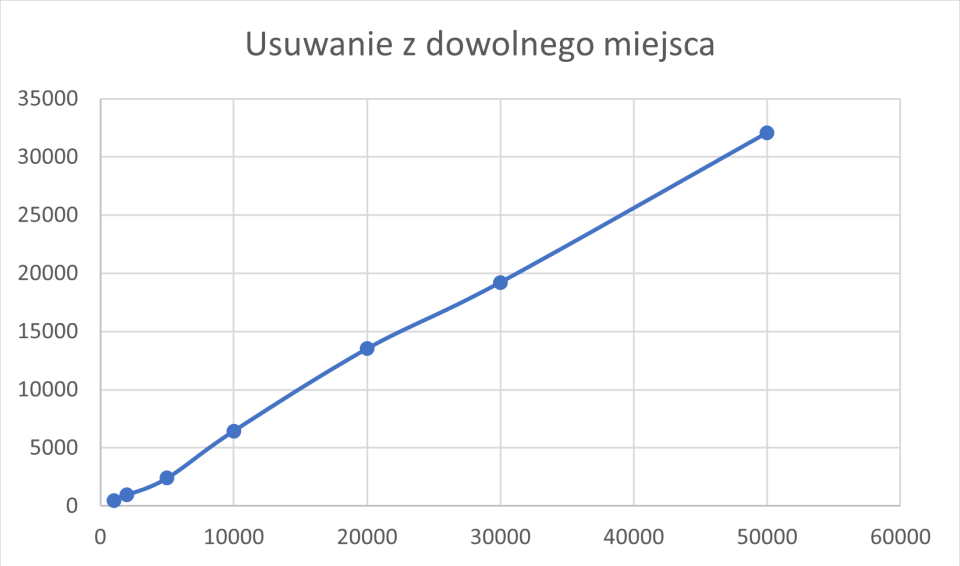


Rysunek 10: Usuwanie z końca w liście

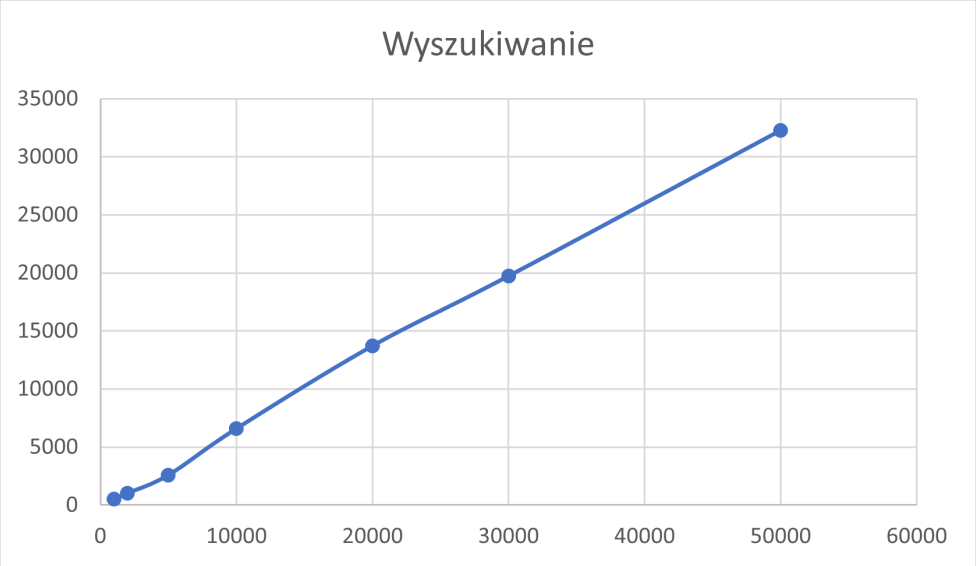


Rysunek 11: Dodawanie na końcu w liście

Rysunek 12: Dodawanie w dowolne miejsce w liście



Rysunek 13: Usuwanie z dowolnego miejsca w liście



Rysunek 14: Wyszukiwanie w liście

3.4 Wnioski

Operacje dodawania i usuwania w liście są bardzo efektywne ze złożonością O(1). W porównaniu do tablicy, potrzebują znacznie mniej czasu. W przypadku, kiedy potrzebne jest wyszukiwanie, lista jest niemal 4 razy mniej efektywna i rośnie liniowo. Można więc wywnioskować, że lista jest dobrą strukturą, kiedy często zachodzi potrzeba dodawania/usuwania elementów, podczas gdy przy wyszukiwaniu lepiej sprawuje się tablica.

4 Kopiec

4.1 Opis

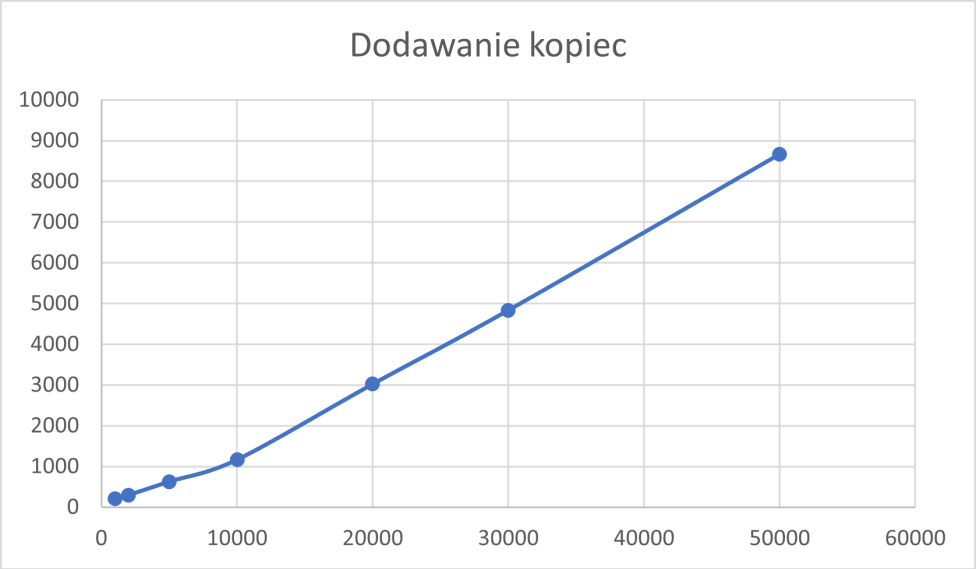
Kopiec maksymalny jest strukturą danych będącą kompletnym drzewem binarnym. Wszystkie poziomy za wyjątkiem ostatniego są zapełnione, natomiast ostatni poziom jest zapełniany od lewej do

prawej. W kopcu zaimplementowano metody dodawania, usuwania korzenia oraz wyszukiwania elementu. Kopiec jest zapisany w postaci tablicy dynamicznej. Metoda wyszukiwania polega na sprawdzeniu kopca od lewej do prawej strony.

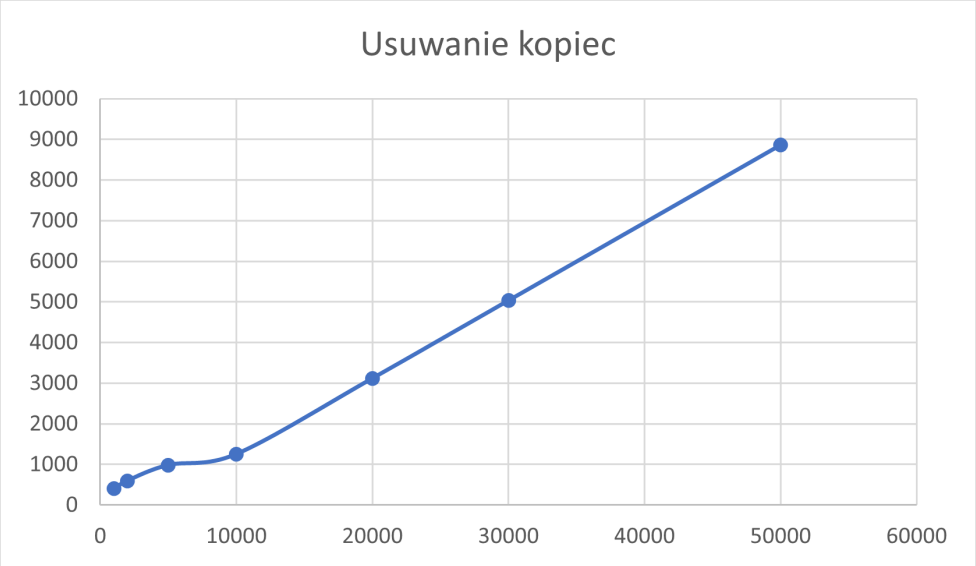
4.2 Pomiary

Ilość elementów	Dodawanie	Usuwanie	Wyszukiwanie
1000	205,3304	398,9002	829,3714933
2000	297,6738	598,8204667	1591,713571
5000	625,2448667	982,8955333	4913,423333
10000	1166,096667	1252,396667	8235,089333
20000	3022,549333	3119,037333	16096,00769
30000	4837,153333	5035,636	17260,28
50000	8671,976667	8865,914667	44955,82308

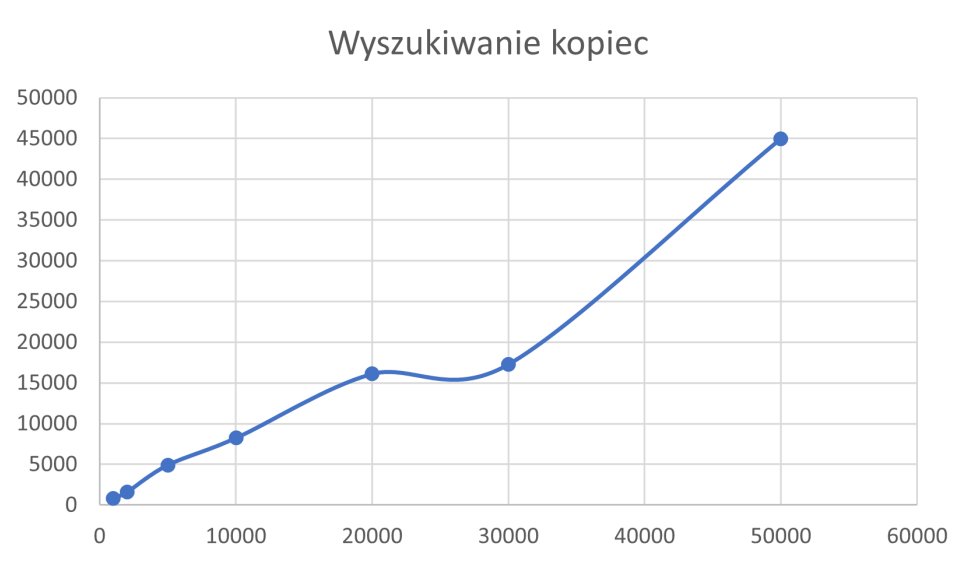
4.3 Wykresy



Rysunek 15: Dodawanie w kopcu



Rysunek 16: Usuwanie w kopcu



Rysunek 17: Wyszukiwanie w kopcu

4.4 Wnioski

Operacje wykonywane na kopcu zabierają bardzo dużo czasu i wszystkie mają liniową złożoność. Jako, że dane kopca są przechowywane przez tablicę dynamiczną, dodawanie i usuwanie wymagają relokacji przy każdym przywołaniu. Pomimo implementacji specjalnego algorytmu do wyszukiwania, ta operacja w kopcu jest nadal bardzo czasochłonna. Kopiec wydaje się świetną strukturą do wyznaczania maksymalnego/minimalnego elementu.

5 Drzewo czerwono-czarne

5.1 Opis

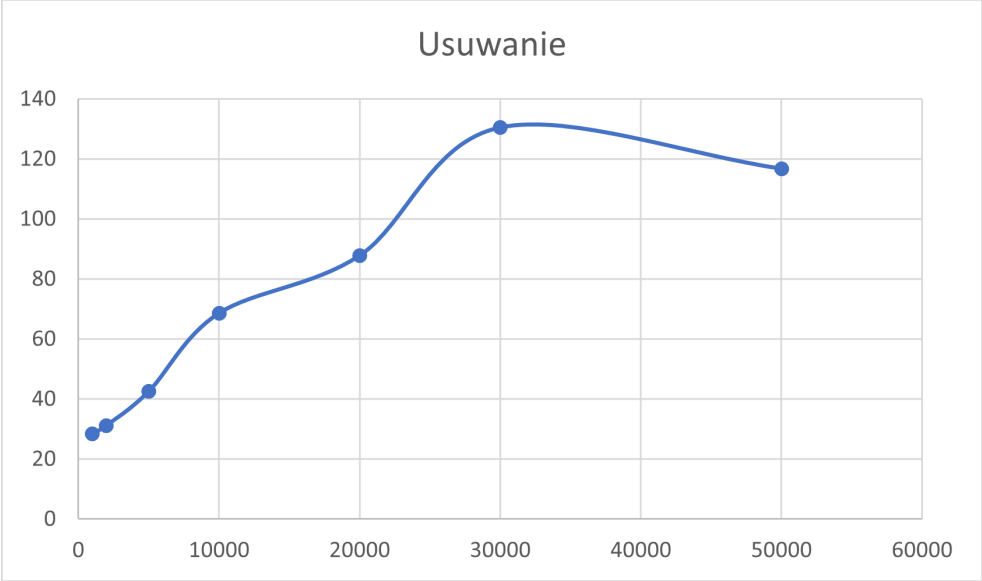
Drzewo czerwono czarne jest odmianą samoorganizujących się drzew BST. W liściach drzewa nie przechowuje się danych, ale wartości NULL. Są to tzw. strażnicy. Założenia drzewa sprawiają, że

jego wysokość nie przekroczy dwukrotnej wartości wysokości minimalnej. Dokonywane jest to przez kolorowanie węzłów na czerwono lub czarno i stosowanie po każdej operacji wstawiania lub usuwania odpowiedniej procedury równoważącej drzewo.

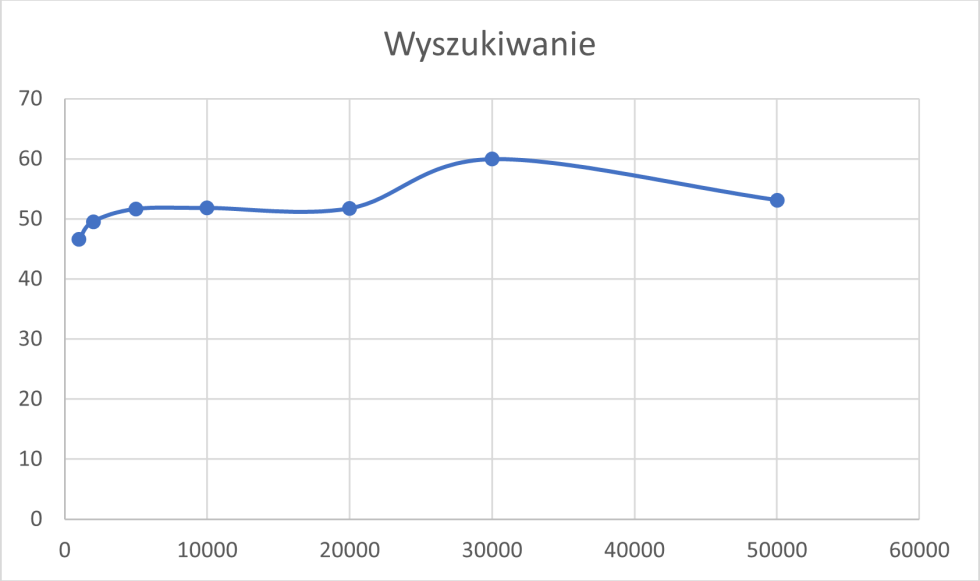
5.2 Pomiary

Ilość elementów	Dodawanie	Usuwanie	Wyszukiwanie
1000	58,888	28,37310667	46,56045333
2000	55,59000667	31,17511333	49,53333333
5000	54,74026	42,47822	51,6351
10000	49,08266	68,568	51,79733333
20000	50,22555333	87,78778	51,72576667
30000	82,67535333	130,47438	59,93377333
50000	79,39532	116,7062	53,10309333

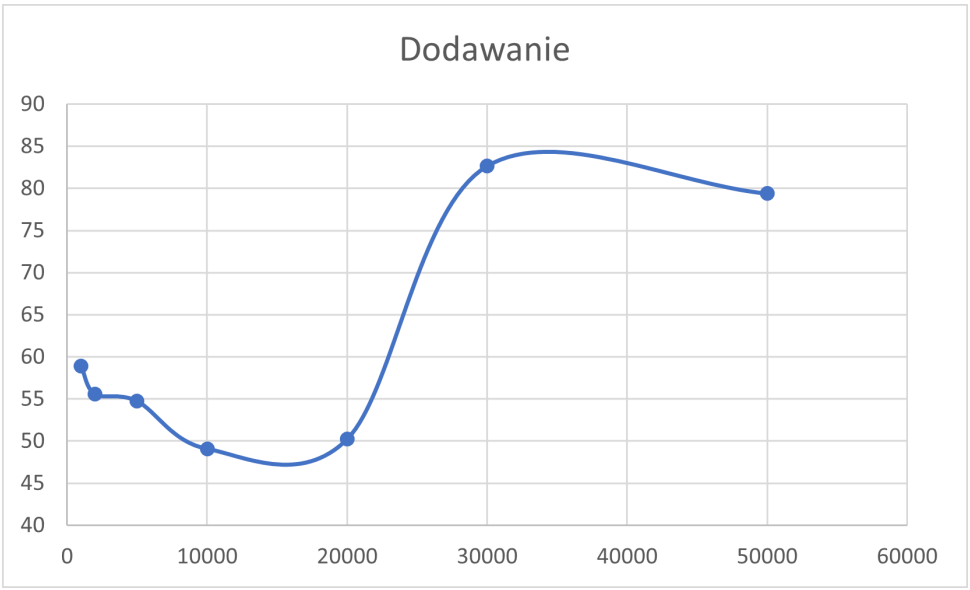
5.3 Wykresy



Rysunek 18: Usuwanie w drzewie czerwono-czarnym



Rysunek 19: Wyszukiwanie najmniejszego elementu w drzewie czerwono-czarnym



Rysunek 20: Dodawanie w drzewie czerwono-czarnym

5.4 Wnioski

Operacje zaimplementowane w drzewie czerwono-czarnym zajmują bardzo mało czasu. Biorąc pod uwagę jednostki zastosowane przy zbieraniu danych do pomiarów, można założyć, że różnice pomiędzy wynikami są na tyle małe, że mają złożoność O(1). Wyszukiwanie w drzewie czerwono-czarnym jest bardzo efektywne i bezkonkurencyjne na tle pozostałych struktur.