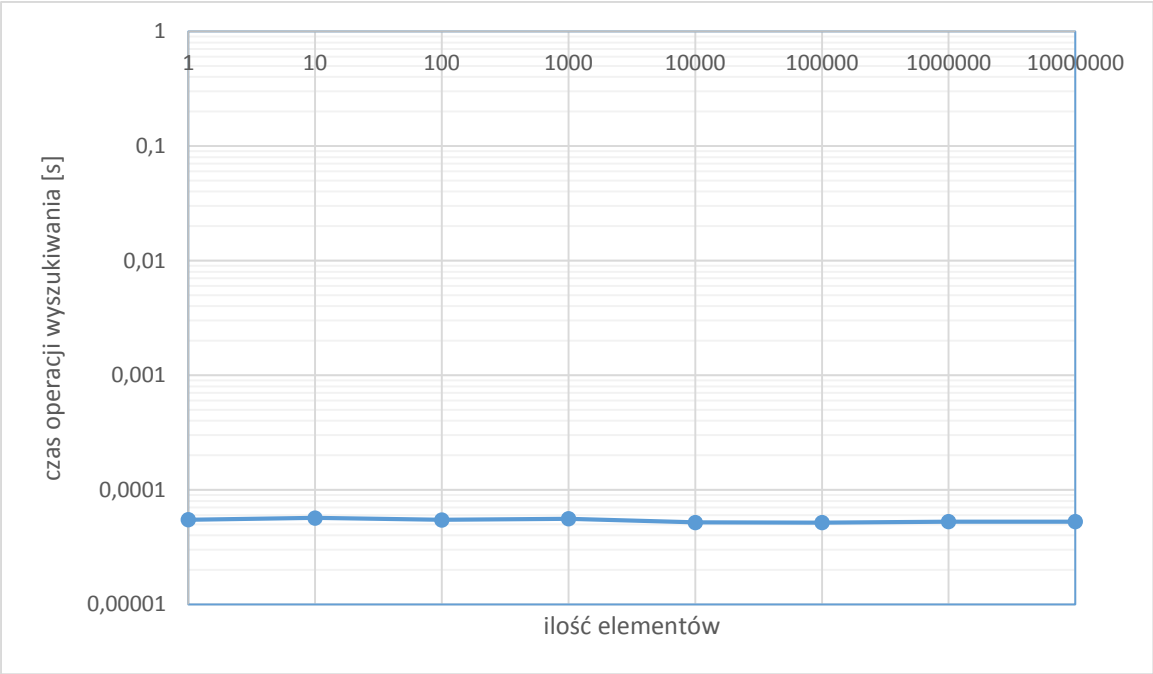
Sprawozdanie

Implementacja Drzewa binarnego samoczynnie się równoważącego

Zadanie polegało na zaimplementowaniu binarnego drzewa poszukiwań. W takiej strukturze lewe poddrzewo każdego węzła zawiera wyłącznie elementy o kluczach mniejszych niż klucz węzła, a prawe poddrzewo zawiera wyłącznie elementy o kluczach nie mniejszych niż klucz węzła. Koszt wykonania podstawowych operacji na takiej strukturze czyli wstawienie, wyszukanie, usunięcie węzła jest proporcjonalny do wysokości drzewa *h*, ponieważ operacje wykonywane są wzdłuż drzewa. Jest to informacja najbardziej kluczowa, mająca największy wpływ na złożoność obliczeniową operacji. Gdy drzewo binarne jest zrównoważone, tzn. każdy węzeł oprócz końcowych liści ma swoich dwóch synów to złożoność obliczeniowa operacji wyszukania klucza jest równa $O(\log_2 n)$. Najbardziej patologiczna sytuacja może wystąpić, gdy struktura drzewa binarnego przypomina strukturę listy, wtedy złożoność obliczeniowa operacji wynosi O(n). Jak widać różnica jest ogromna, dlatego też w drzewie potrzebne są metody na bieżąco go równoważące. Na te potrzeby do implementacji wybrałem drzewo AVL. Poniżej znajduje się tabela wyników wyszukiwania elementu drzewa znajdującego się na samym jego końcu.

Ilość elementów w drzewie	Czas wyszukania klucza[s]
1	0,000054828
10	0,00005686
100	0,000054752
1000	0,000055883
10000	0,00005197
100000	0,000051722
1000000	0,000052817
10000000	0,000052733

Poniżej znajduje się wykres prezentujący dane z tabeli.



Jak widać na wykresie czasy te są porównywalne, jest to spowodowane tym, że dla przypadku w którym w drzewie jest tylko 1 element, funkcja wykona jedno porównanie. Natomiast, gdy w drzewie jest 10^7 elementów funkcja ta wykona zaledwie maksymalnie 24 porównań, by znaleźć element. Są to czasy tak znikome, że najmniejsze zakłócenie działających w tle aplikacji ma ogromny wpływ na wynik pomiaru. Niestety przyrost zużywanej pamięci rezerwowanej dla kolejnych elementów drzewa rośnie liniowo ze wzrostem ilości tych elementów, dlatego też nie można było zmierzyć czasu wyszukiwania dla 10^8 i więcej elementów, a uzyskany wykres w niczym nie przypomina zależności $\log_2 n$.