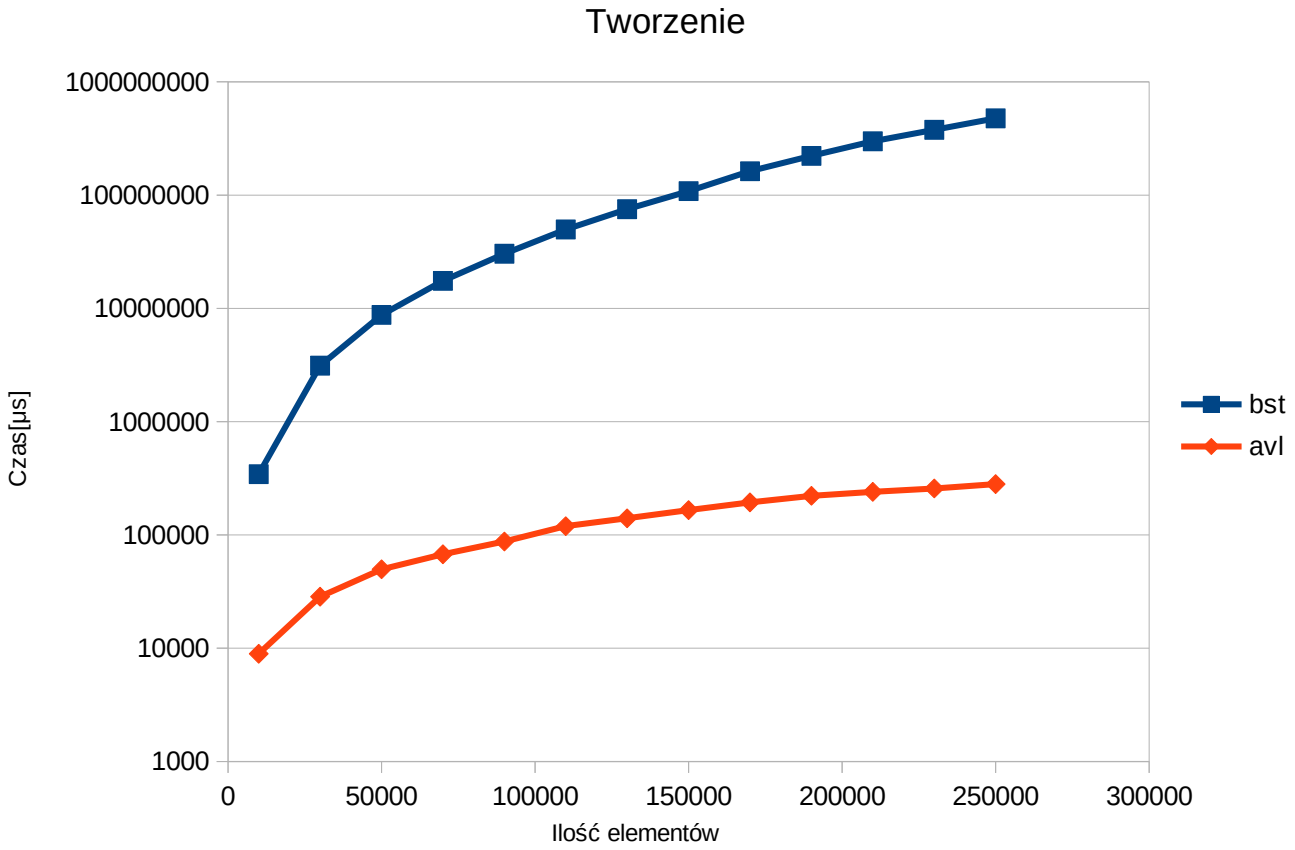
Algorytmy i struktury danych – Drzewa

Filip Rosiak

Pomiary wykonywane były na tablicach posortowanych malejąco o wielkości n od 10 000 elementów do 250 000 elementów, z krokiem 20 000 (13 punktów pomiarowych).

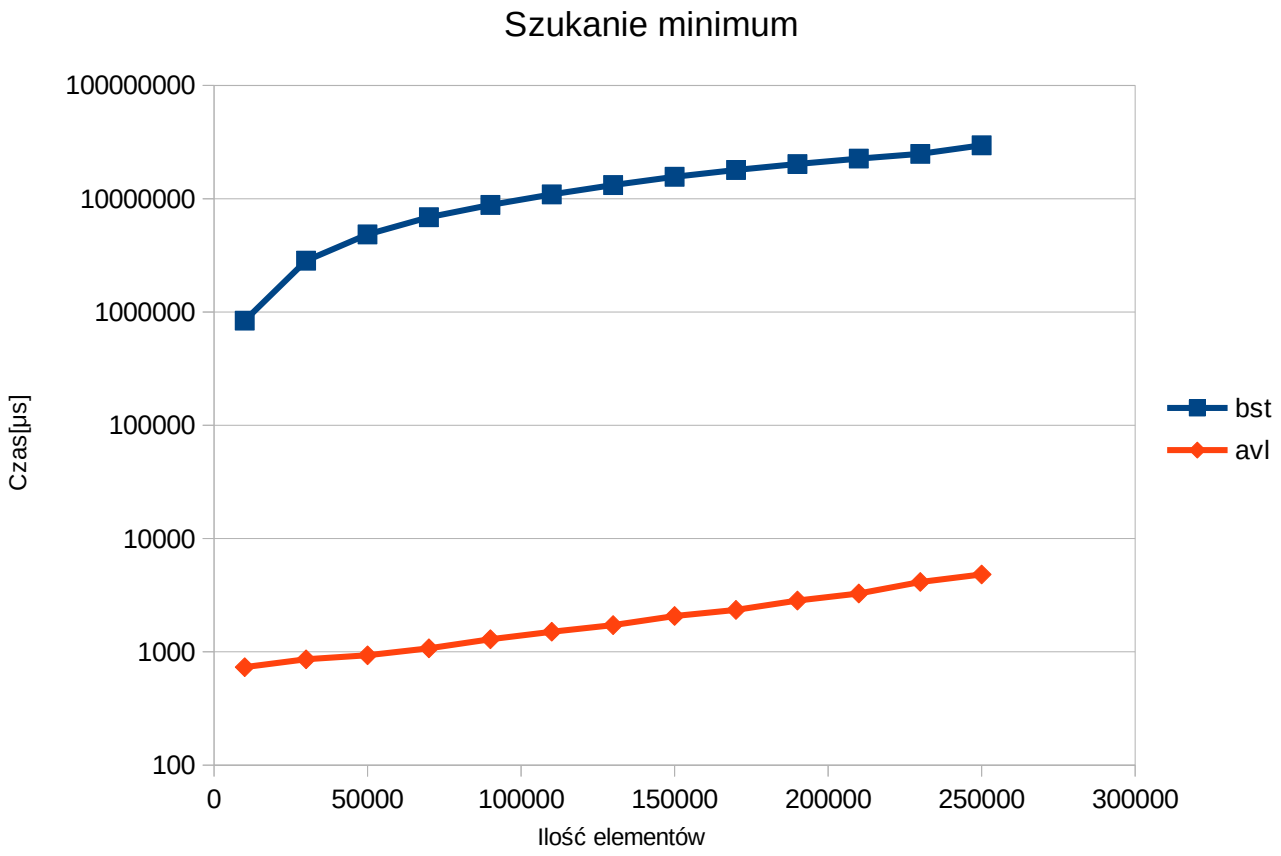
1. Wykresy zależności czasu obliczeń od liczby elementów w drzewie dla różnych operacji

a) Tworzenie drzewa



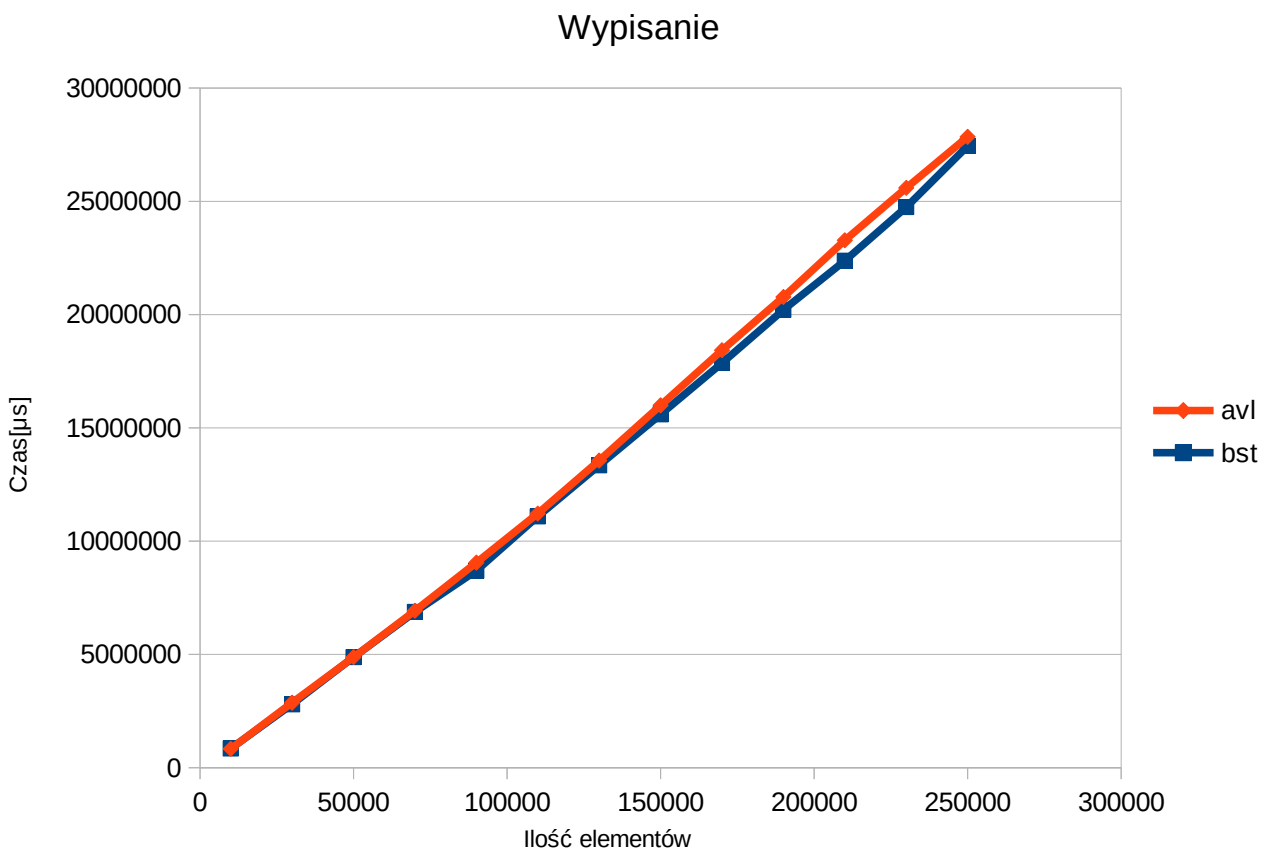
Pod względem czasu potrzebnego do utworzenia struktury, drzewo AVL wypada znacznie lepiej niż drzewo BST. Czynność dodawania n elementów ma złożoność $O(n^2)$ dla drzewa BST. Spowodowane jest to formą tablicy wejściowej, elementy są dodawane malejąco (przypadek zdegradowany). Dla losowej tablicy operacja ta ma w średnim przypadku złożoność O(log n). Drzewo AVL z kolei, dodając kolejne elementy układa je w taki sposób, aby kolejne poddrzewa miały prawie taką samą wysokość. Powoduje to, że złożoność dodania n elementów spada do O(n*log n).

b) Wyszukiwanie minimum



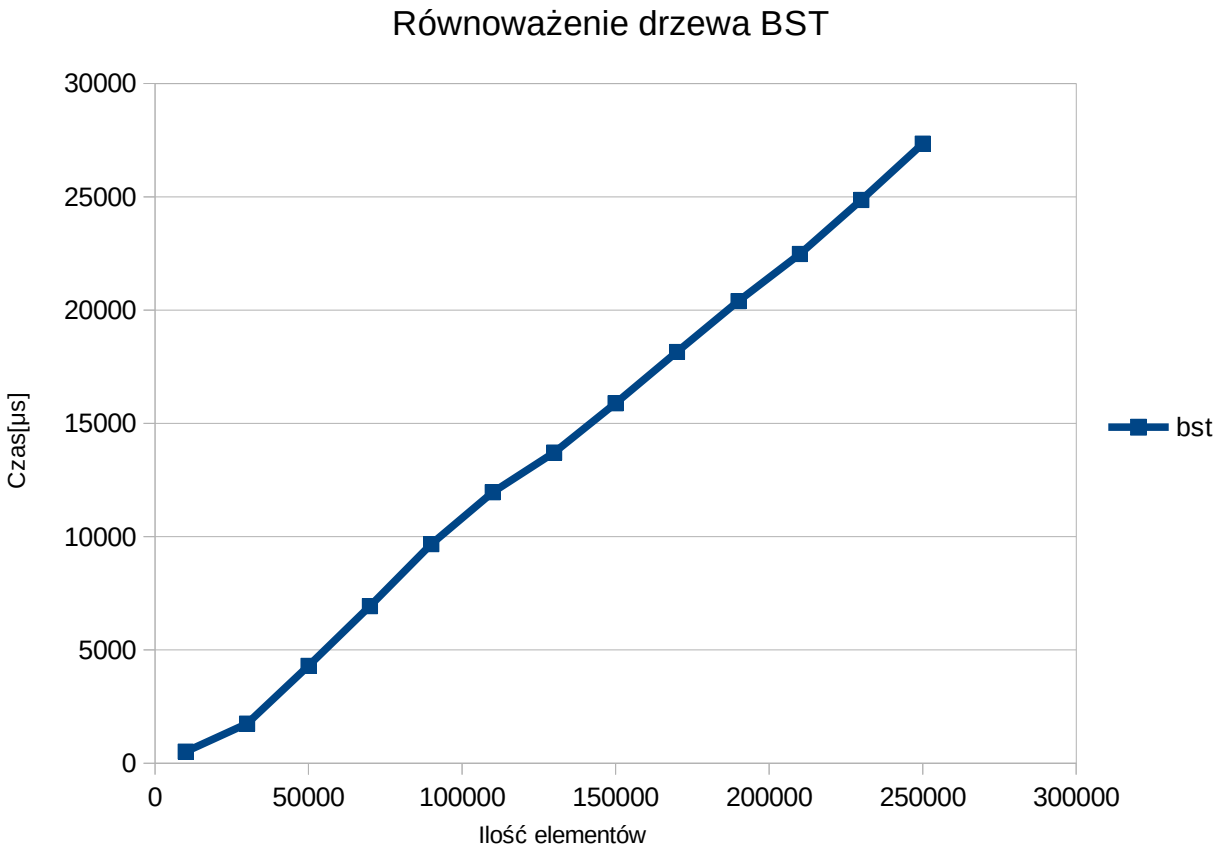
W wyszukiwaniu znów lepiej wypada drzewo AVL. W przypadku szukania najmniejszego elementu, zdegradowane drzewo BST musi udać się do jedynego liścia, który jest na samym końcu tej struktury, co sprawia, że złożoność tej operacji wynosi O(n). Drzewo AVL z kolei, musi jedynie dojść do liścia znajdującego się na wysokości log n, co powoduje, że ta operacja ma złożoność O(log n).

c) Wypisanie in-order



W przypadku wypisania in-order, nie widać zbyt dużej różnicy w czasach. Algorytm wypisania in-order ma złożoność O(n).

2. Wykres zależności czasu balansowania drzewa BST od ilości elementów



Balansowanie zostało wykonane za pomocą algorytmu DSW, który na początku tworzy winorośl z drzewa. Następnie drzewo zostaje poddane kompresji poprzez serię operacji przekręcenia danego korzenia w lewo. Cały algorytm ma złożoność O(n).