ZŁOŻONE

STRUKTURY DANYCH

Wykonanie:

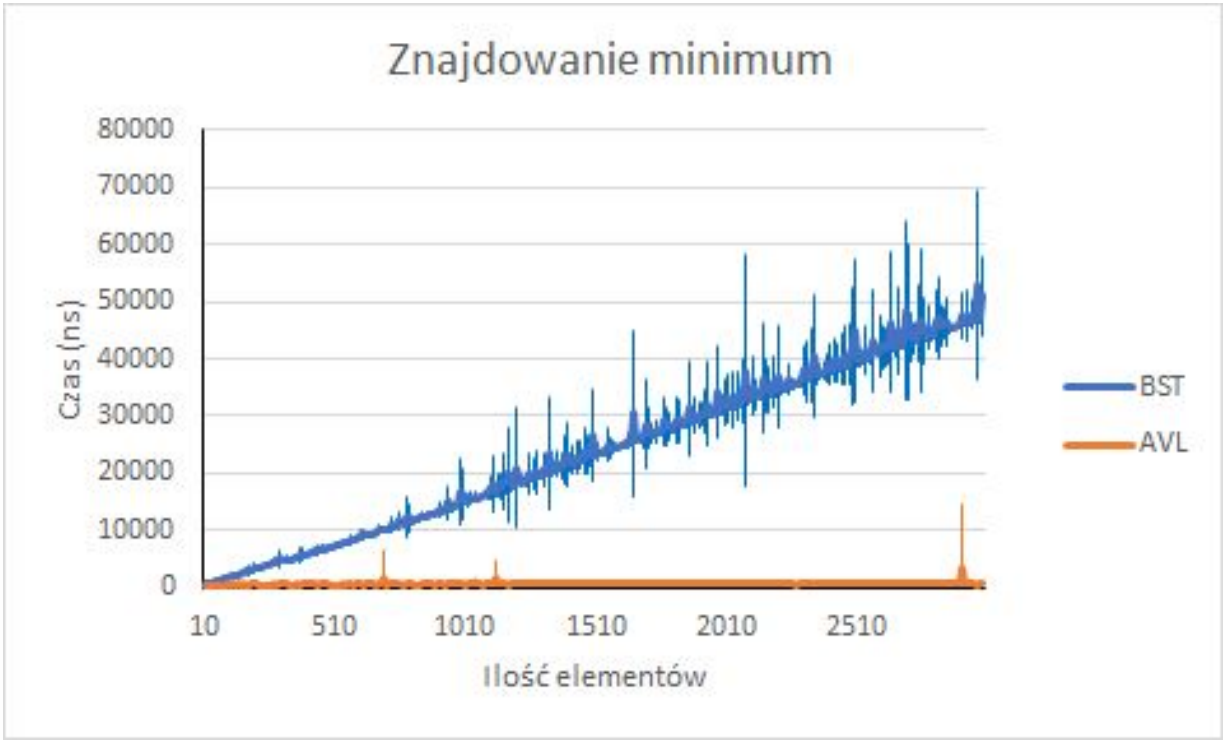
Adrian Madajewski 145406

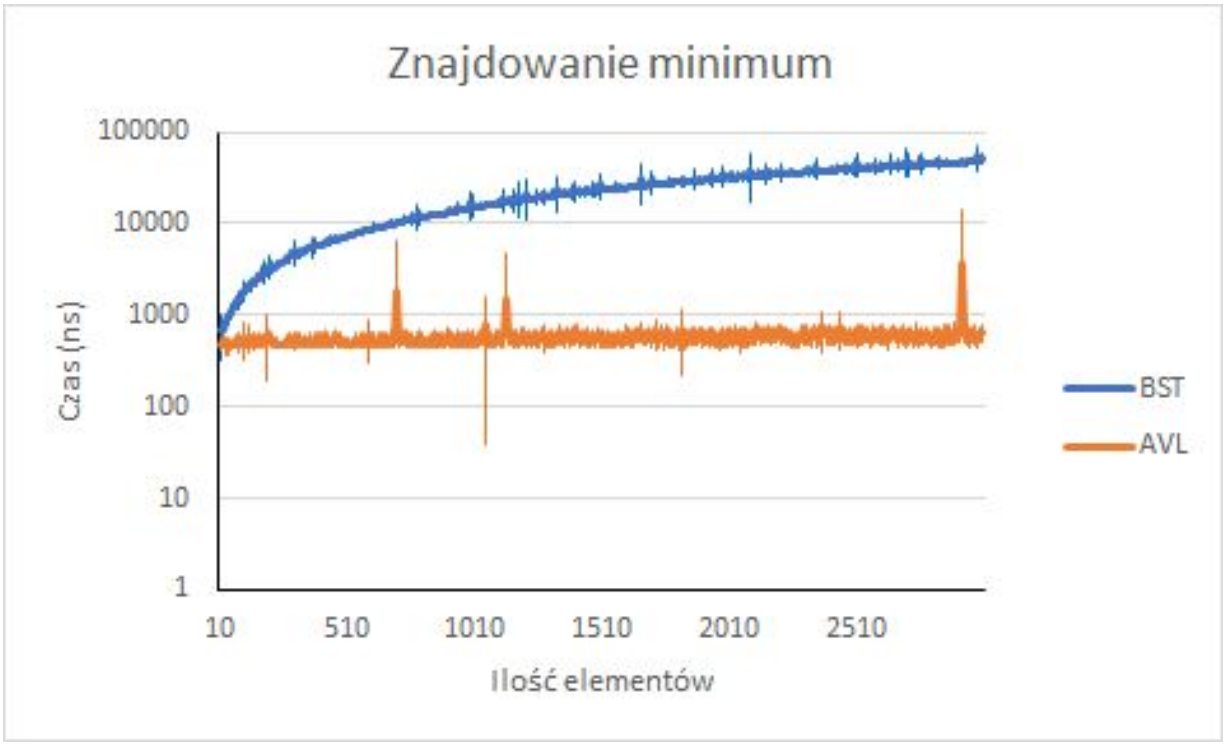
Michał Kwarta 145192

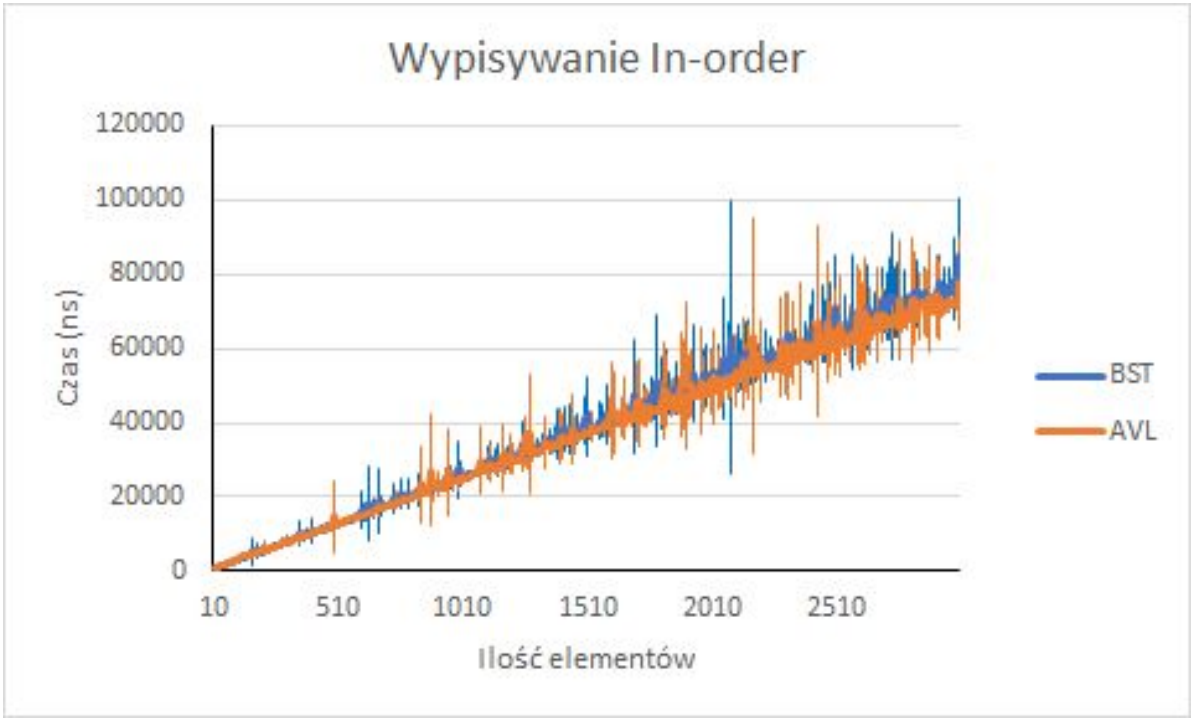
Wykresy czasu od ilości elementów dla wybranych procedur

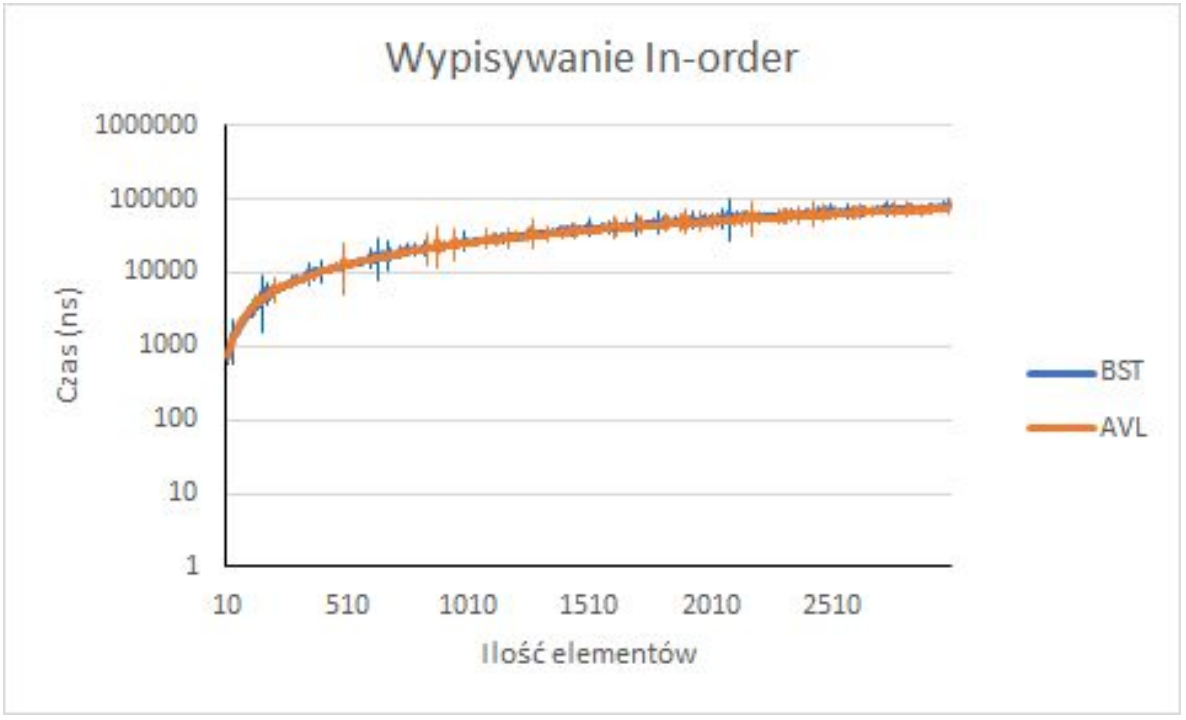
Poniżej przedstawione są wykresy w skali liniowej i logarytmicznej. Są one wynikiem 15 testów. Pierwsza liczba ciągu była losowo wybierana z przedziału $[6000, 12000]$, a każda następna była równa różnicy liczby poprzedniej i losowo wygenerowanej liczby z przedziału $[1, 100]$. Dostęp do arkusza z danymi znajduje się na końcu dokumentu.

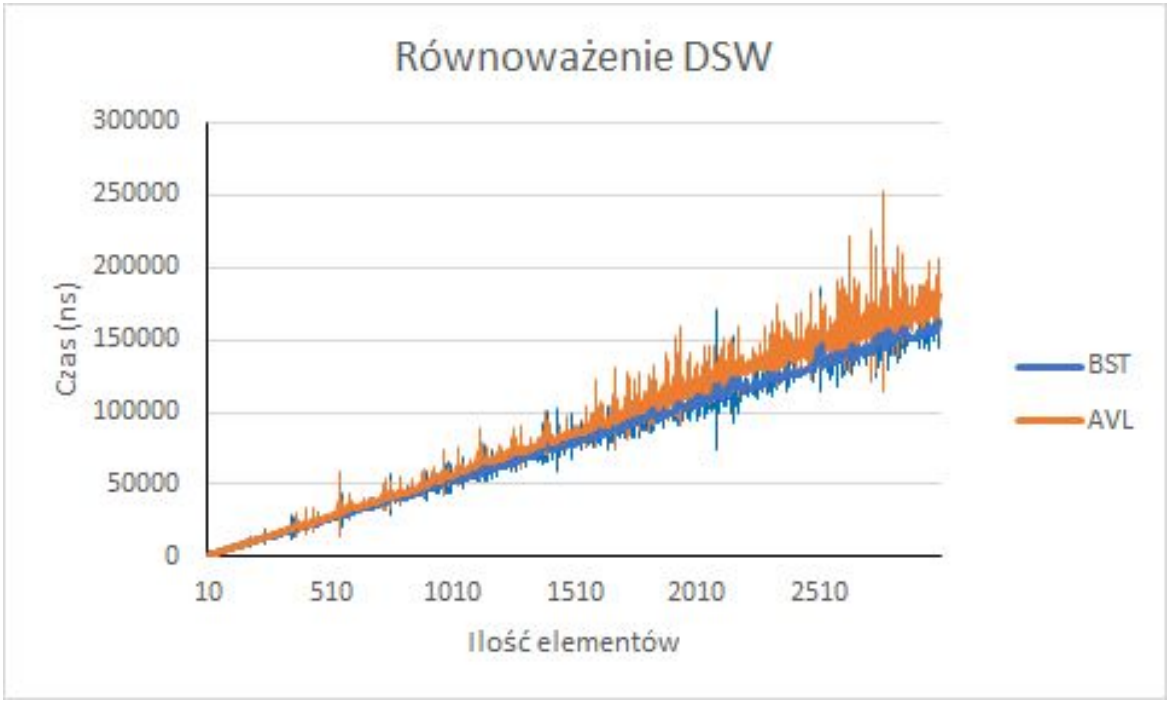

Tworzenie drzewa
100000000
10000000
1000000
100000
10000
1000
100
10
1
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL

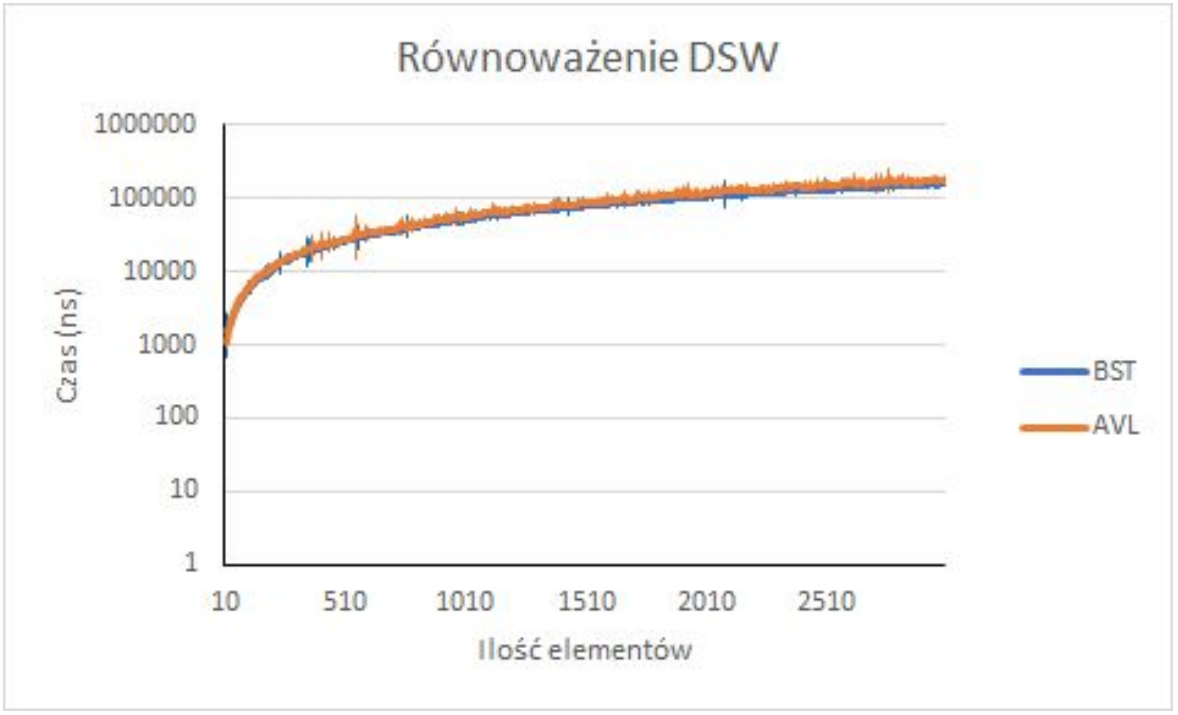

Znajdowanie minimum
80000
70000
60000
50000
40000
30000
20000
10000
0
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL


Znajdowanie minimum
100000
10000
1000
100
10
1
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL


Wypisywanie In-order
120000
100000
80000
60000
40000
20000
0
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL


Wypisywanie In-order
1000000
100000
10000
1000
100
10
1
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL


Równoważenie DSW
300000
250000
200000
150000
100000
50000
0
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL


Równoważenie DSW
1000000
100000
10000
1000
100
10
1
Czas (ns)
10
510
1010
1510
2010
2510
Ilość elementów
BST
AVL

Wnioski

Generowanie drzewa

Złożoność obliczeniowa tworzenia dowolnego drzewa jest mocno zależna od rozkładu danych, co widać na wykresie. Aby utworzyć drzewo *AVL* trzeba umieścić n elementów, a złożoność operacji *insert* wynosi w drzewie *AVL* zawsze $O(log_2 n)$, a więc cała procedura generowania drzewa *AVL* ma złożoność $O(n \cdot log_2 n)$. Drzewo *BST* według założeń zadania ma być generowane z ciągu malejącego. Taki posortowany ciąg sprawia, że złożoność operacji *insert* jest najgorsza możliwa,dlatego, że umieszczenie kolejnego elementu wiąże się z porównaniem go z każdym już znajdującym się w drzewie, a więc złożoność wynosi $O(n)$, co za tym idzie umieszczenie n elementów sprowadza złożoność generowania drzewa do $O(n^2)$

Znajdowanie minimum

W przypadku drzewa *BST* złożoność obliczeniowa może być skrajnie różna, dlatego że jest bezpośrednio powiązana z doborem korzenia, a sama w sobie zawsze równa jest wysokości drzewa. Dla przykładu jeśli drzewo będzie utworzone z kolejnych elementów ciągu $[1, 2, 3, ..., n]$ będzie to optymistyczny przypadek, w którym korzeniem drzewa będzie liczba najmniejsza, więc znalezienie minimum to będzie jedynie jedna operacja, a samo drzewo będzie drzewem zdegenerowanym (winoroślą), ale w dokładnie odwrotnym przypadku czyli biorąc kolejno elementy z listy $[n, ..., 3, 2, 1]$ wtedy złożoność ta będzie wynosiła $O(n)$, jednakże złożoność tej procedury uśrednia się do $O(log_2 n)$.

W drzewie *AVL* nie ma aż takiej rozbieżności, drzewo AVL jest zawsze zrównoważone więc ilość kolejnych elementów do sprawdzenia jest zawsze równa h lub $h-1$, a więc złożoność zawsze sprowadza się do $O(h) = O(log_2 n)$

Wypisywanie in-order

Procedura wywołuje się rekurencyjnie dla każdego węzła, raz dla lewego dziecka i raz dla prawego, a więc faktycznie wywołuje się raz dla każdego elementu, inaczej mówiąc złożoność tej procedury jest liniowa, a określa się ją jako $O(n)$.

Algorytm Day-Stout-Warren - *DSW*

Algorytm *DSW* jest wydajną metodą równoważenia drzew, inaczej mówiąc sprowadzania drzewa do postaci o najmniejszej możliwej wysokości. Złożoność tego algorytmu wynosi $O(n)$ i pracuje on w miejscu. Algorytm składa się z dwóch faz: Pierwsza, w której wychodzimy od korzenia i dopóki posiada lewych synów, dokonujemy obrotu w prawo. Założenia naszego zadania są niezwykle niekorzystne dla tego algorytmu, ponieważ lewych synów będzie $n-1$, co za tym idzie, trzeba przejść przez $2n-1$ węzłów i dokonać $n-1$ rotacji. W przypadku optymistycznym byłoby to przejście n węzłów bez rotacji. Oba przypadki mają złożoność $O(n)$. Druga faza w której za pomocą obrotów w lewo na co drugim węźle przekształcamy listę liniową w drzewo.
Sprowadza się to do $m - log(m+1)$ iteracji pętli i $n - \lfloor log(n+1) \rfloor$ obrotów, gdzie $m = 2^{\lfloor log(n+1) \rfloor} - 1$. Obie fazy mają złożoność $O(n)$, a więc złożoność całego algorytmu również jest równa $O(n)$.

Informacje dodatkowe

link do arkusza z danymi
strona github z kodem źródłowym

Platforma testowa

procesor:	Intel i7 7700k 4.2 GHz
ram:	8 GB RAM DDR4 CL16
karta graficzna:	Geforce GTX 1060 6GB
system operacyjny:	Windows 7 64-bit
płyta główna:	MSI Z270-A PRO
dysk:	HDD 1000 GB

Źródła dodatkowe

Introduction to Algorithms Thomas H. Cormen
eduinf.waw.pl