Prowadzący:
dr inż. Marcin Szlenk

27.11.2021r.

ALGORYTMY
I
STRUKTURY DANYCH

Laboratorium
Drzewa

Wykonali:
Igor Nowak
Przemysław Wiszniewski

1. Cel projektu.

Celem projektu było przygotowanie implementacji drzewa BST (ang. *Binary Search Tree*) i drzewa AVL wraz z operacjami wstawiania, usuwania, wyszukiwania elementu oraz wyświetlania całego drzewa na ekranie. Następnie zbadaliśmy czas wykonywania tych operacji dla podanej tablicy wejściowej zawierającej liczby losowane z zadanego zakresu.

2. Środowisko programistyczne i platforma testowa.

Algorytmy oraz skrypt do wykonywania pomiarów zostały zaimplementowane w języku Python. Na platformę testową składał się komputer wyposażony w procesor AMD Ryzen 5 3600 taktowany zegarem 4.2 GHz operujący pod systemem Windows 10 Pro w wersji 21H1 oraz środowisko języka Python w wersji 3.9.4.

3. Pliki i funkcje.

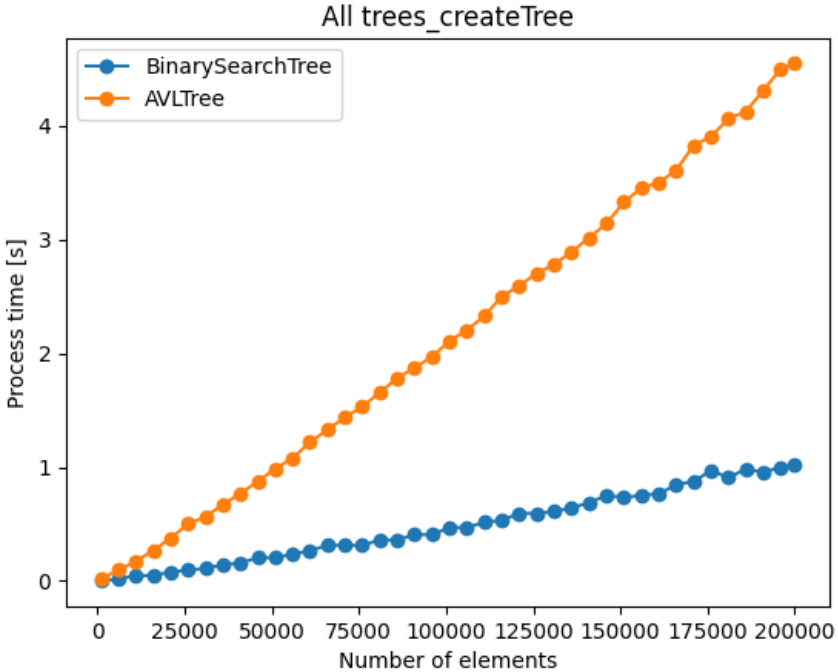
Skrypt testujący został podzielony na kilka plików składowych:

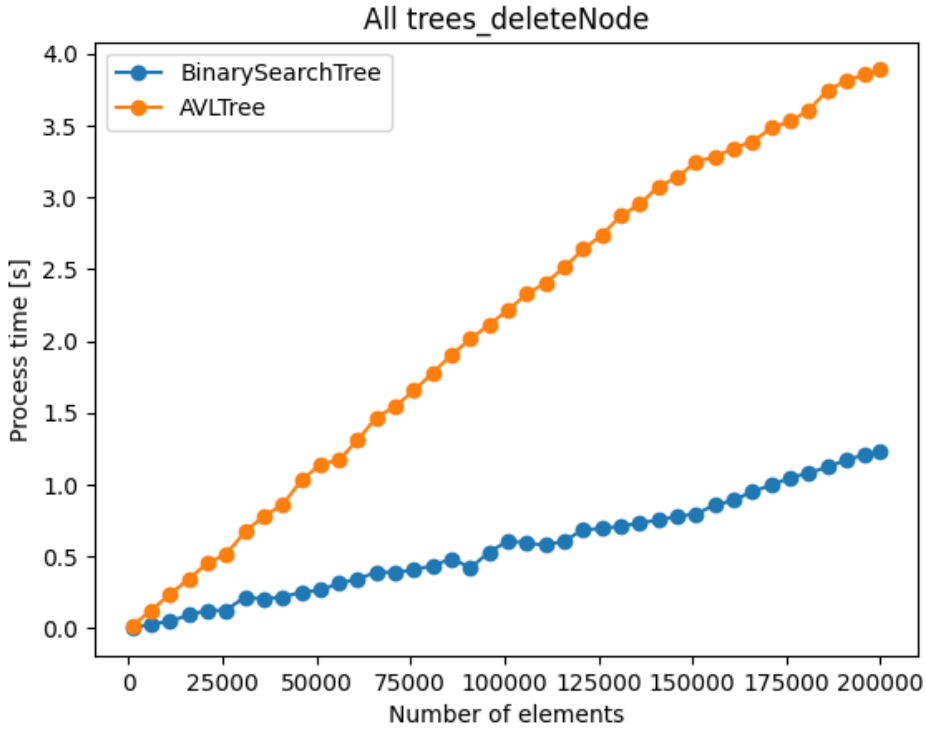
- **Main.py** – główny plik zawierający tylko jedną funkcję odpowiedzialną za wystartowanie pomiarów czasu dla zadanej listy zawierającej tablice elementów, na których będą wykonywane pomiary.
- **Utils.py** – plik zawierający klasę Utils, odpowiedzialną za obsługę wszystkich czynności niezbędnych do wykonania pomiarów, takich jak wygenerowanie danych, wystartowanie pomiarów, stworzenie wykresów i ich zapis do pliku oraz zapis wyników pomiarów do pliku.
- **Node.py** – plik zawierający klasę Node implementującą węzeł, z których zbudowane są drzewa.
- **BinarySearchTree.py** – plik zawierający klasę BinarySearchTree implementującą drzewo BST.
- **AVLTree.py** – plik zawierający klasę AVLTree implementującą drzewo AVL.

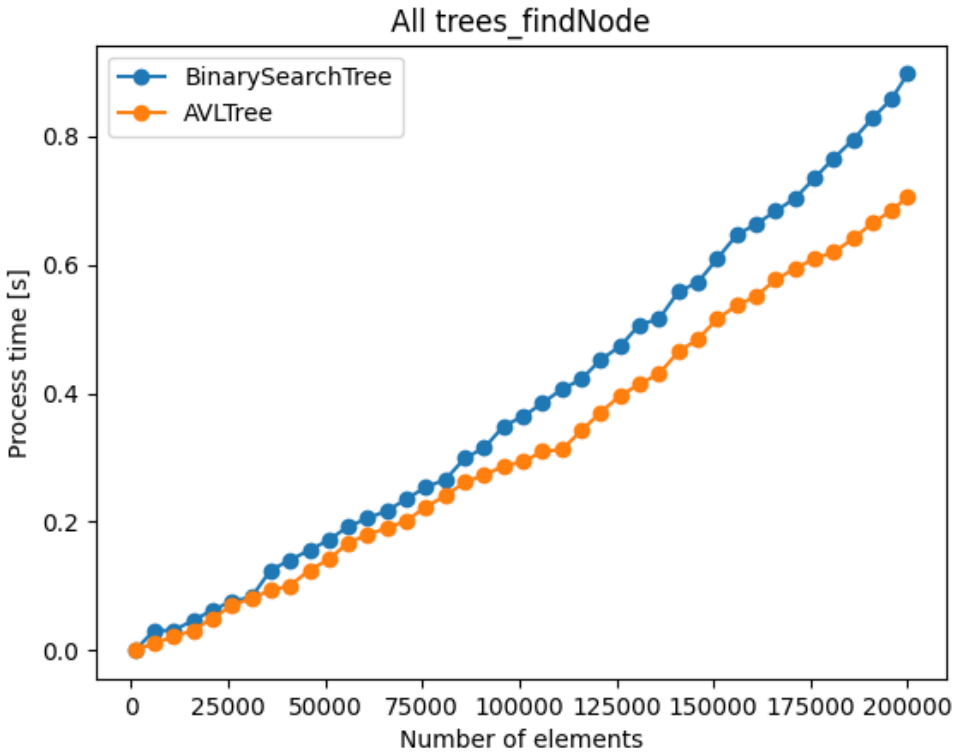
W celu uruchomienia pomiarów należy w konsoli użyć polecenia ***python Main.py***. Aby zmienić wejściowe tablice elementów należy w pliku **Main.py** zmodyfikować zmienną ***list_of_lists***.

4. Wyniki pomiarów.

Pomiary zostały wykonane dla liczby elementów z przedziału 1000 – 200000 z krokiem 5000.




All trees_deleteNode
BinarySearchTree
AVLTree
Process time [s]
Number of elements
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
25000
50000
75000
100000
125000
150000
175000
200000


All trees_findNode
BinarySearchTree
AVLTree
Process time [s]
Number of elements
0.0
0.2
0.4
0.6
0.8
0
25000
50000
75000
100000
125000
150000
175000
200000

5. Podsumowanie.

Analizując wykresy widzimy, że tworzenie drzewa AVL oraz usuwanie z niego elementów trwa znacznie dłużej niż drzewa BST. Jest to spowodowane tym, iż w drzewie AVL po każdej operacji dodania lub usunięcia elementu należy wykonać procedurę balansowania drzewa. Balansowanie drzewa sprawia, że staje się on równomiernie rozłożone. Dzięki temu wyszukiwanie elementów staje się szybsze, co potwierdzają wykresy z wykonanych pomiarów. Wraz ze wzrostem ilości wykonywanych wyszukiwań różnica czasów ich wykonania dla drzewa AVL i BST staje się coraz większa na korzyść drzewa AVL.