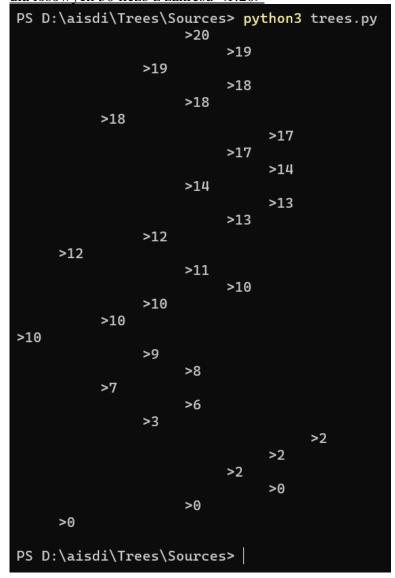
AISDI lab.2 Drzewa

Zaimplementowane zostały klasy reprezentujące 2 rodzaje drzew BST i AVL, oraz klasa reprezentująca węzeł drzewa. Węzły mają określone parametry takie jak przodek oraz lewy i prawy potomek.

W klasie BST występują następujące metody:

- <u>znajdowanie węzła o danej wartości w drzewie</u> dajemy wskaźnik na korzeń drzewa i porównujemy wartość konkretnego węzła z wartością szukaną. Jeśli zachodzi nierówność schodzimy odpowiednio do lewego, lub prawego potomka i powtarzamy operację do momentu, aż nie znajdziemy wartości, lub drzewo się nie skończy.
- wstawianie elementu o danej wartości do drzewa przebiega bardzo podobnie jak szukanie danej wartości w drzewie z tą różnicą, że jak dojdziemy do właściwego miejsac dodajemy do drzewa daną wartość
- <u>usuwanie danego elementu z drzewa j</u>eśli dany element jest liściem robimy to bez zmieniania struktury drzewa. Jeśli ma jednego potomka, wstawiamy go w miejsce usuniętego węzła. Jeśli za to ma dwóch potomków, w jego miejsce wstawiamy następnik, czyli węzeł, który nie ma lewego potomka
- wyświetlanie drzewa w interfejsie tekstowym dla losowych 30 liczb z zakresu <1:20>



W klasie AVL analogicznie:

- <u>znajdowanie węzła o danej wartości w drzewie</u> tak samo jak w BST, jednak tutaj unikamy pesymistycznych scenariuszy
- <u>wstawianie elementu o danej wartości do drzewa</u> podobnie jak w BST, tylko teraz sprawdzamy współczynnik wyważenia i w razie potrzeby dokonujemy rotacji do momentu, aż drzewo odzyska własności drzewa AVL
- <u>usuwanie danego elementu z drzewa –</u> podobnie jak przy usuwaniu w BST, a nastepnie przywracamy odpowiedni współczynnik wyważenia
- wyświetlanie drzewa w interfejsie tekstowym, dodatkowo wyświetla wyważenie danego węzła.

Dla losowych 30 liczb z zakresu <1:20>

```
PS D:\aisdi\Trees\Sources> python3 trees.py
                      >20 0
                >19 0
                      >18 0
           >18 0
                      >18 0
                >18 0
                      >18 0
     >17 -1
                           >16 0
                      >16 0
                           >16 0
                >15 1
                      >15 0
           >14 1
                      >12 0
                >12 0
                      >12 0
>10 0
                      >9 0
                >9 0
                      >9 0
           >9 1
                >8 0
     >8 -1
                >7 -1
                      >5 0
           >2 -1
                            >1 0
                      >1 1
                >0 1
                      >0 0
```

Następnie wykonaliśmy pomiary czasu wykonywania powyższych metod przy pomocy benchmarka dla różnych ilości danych. Pomiary umieściliśmy na wykresach porównujących działanie drzewa BST i AVL. W drzewie AVL wstawianie i usuwanie jest wolniejsze, gdyż wymaga większej ilości operacji, za to proces wyszukiwania w AVL jest szybszy.