Lidia Sobońska 310903
Arkadiusz Kojtek 310746

AISDI

Algorytmy tworzenia i przeszukiwania drzew (B)
Sprawozdanie

Oświadczamy że poniższa praca, będąca podstawą do ocenienia drugich laboratoriów z AISDI została wykonana przez nas samodzielnie.

Biblioteki wykorzystane w projekcie

Do stworzenia naszego projektu korzystaliśmy z następujących modułów:

1. **pyplot** z **Matplotlib,** który został wykorzystany to narysowania wykresów przebiegów czasowych.
2. **sys** służący do przyjęcia informacji o pliku źródłowym czy pliku docelowym z wynikami.
3. **time** wykorzystany do zmierzenia czasu testu każdego z drzew.
4. **random,** który posłużył do wygenerowania zestawu losowych liczb.

Algorytmy tworzenia drzew

Poszczególne kody reprezentujące algorytm tworzenia drzew zostały umieszczone w plikach .py gdzie odpowiednio:
bst.py - kod reprezentujący tworzenie drzewa bst (binary search tree)
avl.py - kod reprezentujacy tworzenie drzewa avl (Adelson-Velsky and Landis)

Podział prac

Kod tworzenia drzewa AVL został napisany przez Lidię Sobońską, natomiast kod BST został napisany przez Arkadiusza Kojtek.

Sposób uruchomienia programu

Plikiem obsługującym każdy z projektów jest **main.py**, uruchomienie projektu polega na wywołaniu pliku **main.py** z poziomu terminala z dopisanymi dwoma opcjonalnymi argumentami - **main.py --outfile [nazwa_pliku] --infile [nazwa_pliku]** reprezentującymi odpowiednio plik, do którego zostaną zapisane przebiegi czasowe oraz plik z opcjonalnymi danymi źródłowymi. Jeżeli plik z danymi źródłowymi nie zostanie podany, tworzenie drzew będzie pracować na losowo wygenerowanym zestawie liczb. Jeżeli natomiast nie zostanie podany plik wyjściowy, to domyślnie przebiegi czasowe zostaną zapisane do pliku **times.tsv**.

Przebieg działania programu

Projekt najpierw próbuje wczytać dane do posortowania z wybranego przez użytkownika pliku, po czym przechodzi do mierzenia czasu tworzenia drzew z wykorzystaniem podanych danych, wyszukiwania w nich elementów oraz usuwania elementów. Czas mierzony jest z użyciem funkcji **process_time()**. Informacje na temat wielkości kolekcji z danymi oraz czasu wykonania konkretnych funkcji są na bieżąco zapisywane do tablic, które następnie wykorzystywane są do utworzenia wykresu z wykorzystaniem obiektu **pyplot**.

Pliki wynikowe

Jako plik wynikowy zostaną utworzone pliki **building.png** zawierający graficzne przedstawienie czasu potrzebnego na stworzenie każdego z drzew, **finding.png** zawierający wykres obrazujący przeszukiwanie każdego z drzew oraz **removing.png** obrazujący czas potrzebny na usunięcie poszczególnych elementów każdego z drzew.

Analiza danych

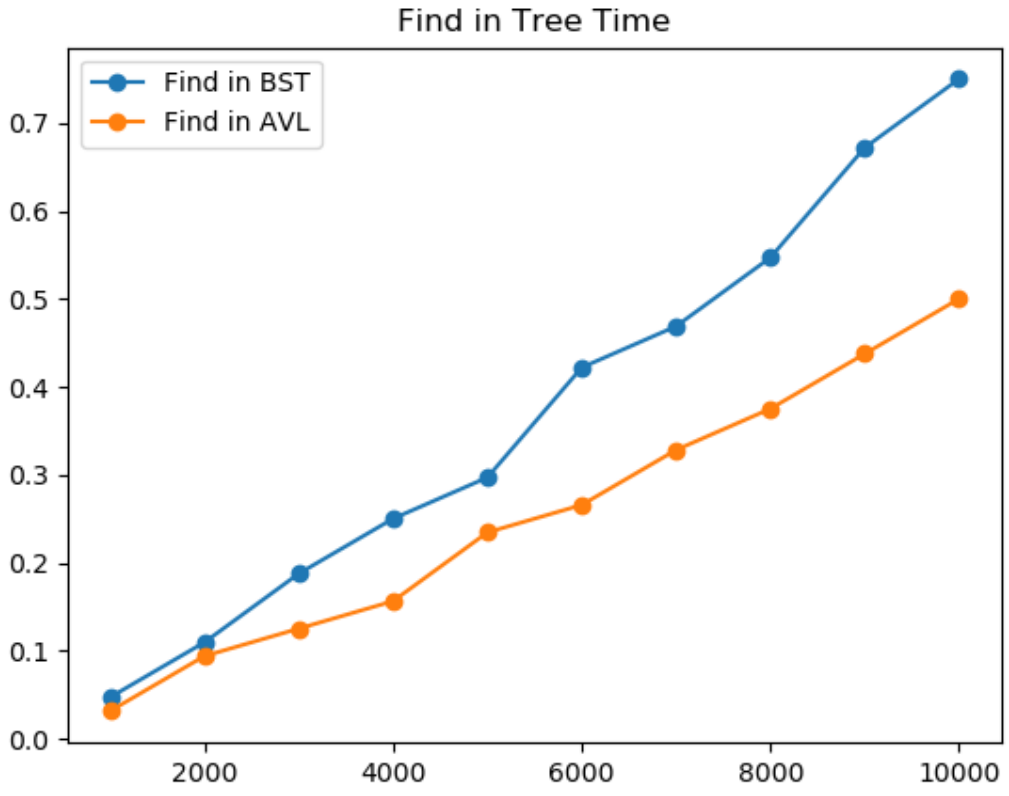
(Wykresy na końcu pliku)

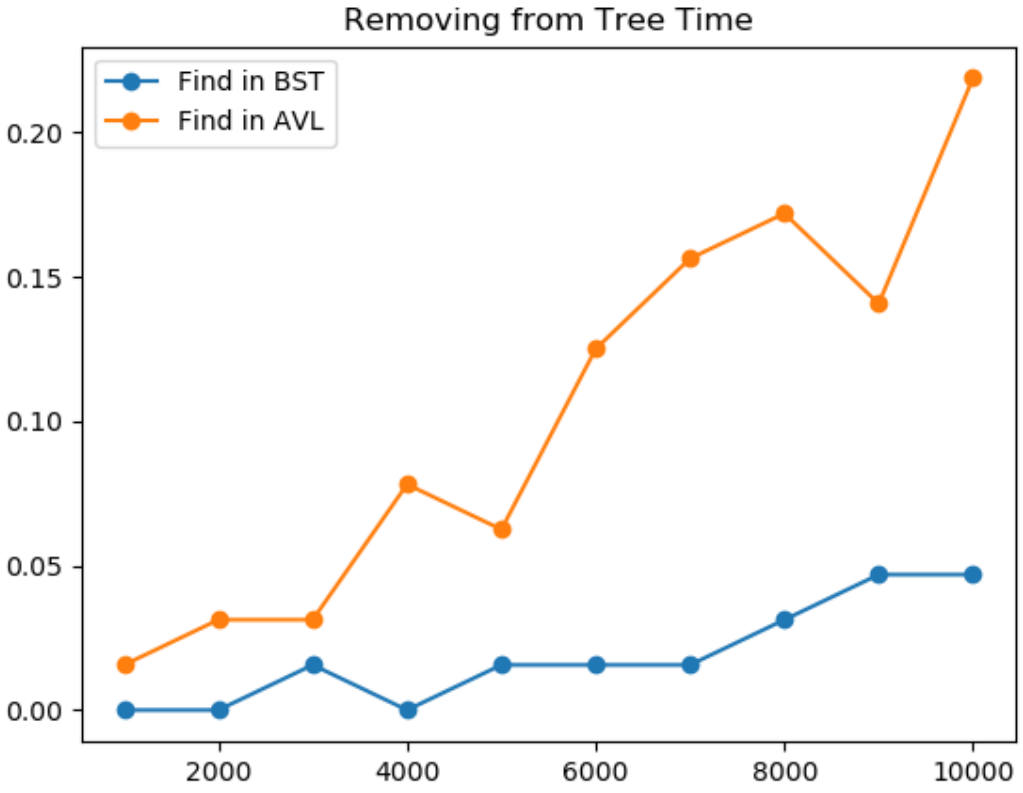
Number of numbers:	Build BST:	Build AVL:	Search in BST:	Search in AVL:	Remove from BST:	Remove from AVL:
1000	0.015625	0.03125	0.046875	0.03125	0.0	0.015625
2000	0.0625	0.0625	0.109375	0.09375	0.0	0.03125
3000	0.015625	0.125	0.1875	0.125	0.015625	0.03125
4000	0.03125	0.15625	0.25	0.15625	0.0	0.078125
5000	0.046875	0.203125	0.296875	0.234375	0.015625	0.0625
6000	0.046875	0.28125	0.421875	0.265625	0.015625	0.125
7000	0.0625	0.328125	0.46875	0.328125	0.015625	0.15625
8000	0.109375	0.328125	0.546875	0.375	0.03125	0.171875
9000	0.09375	0.4375	0.671875	0.4375	0.046875	0.140625
10000	0.09375	0.453125	0.75	0.5	0.046875	0.21875

Uwaga!! Dla niektórych przypadków zmierzony czas jest równy 0s, błąd wynika z niedokładności funkcji użytej do mierzenia czasu poszczególnych testów.

Analiza wykresów oraz tabeli czasów pokazuje przewagę algorytymu BST nad AVL przy tworzeniu oraz usuwaniu elementów z listy. Wynika to z tego, iż w czasie tworzenia czy usuwania drzewo AVL musi być zmieniane, aby zachowywało zrównoważoną strukturę. AVL ma jednak zauważalną przewagę nad BST przy przeszukiwaniu elementów drzewa. Różnica czasów dla tworzenia i usuwania danych, wraz ze zwiększeniem ilości danych jest korzystniejsza dla BST. Tworzenie drzewa oraz usuwanie danych w tym przypadku oscylowało wokół 20% czasu AVL z niewielkimi wahaniami w obie strony. Przy wyszukiwaniu jednak czas BST stanowił około 150% czasu AVL.
Jeżeli używane przez nas drzewo będzie często zmieniane, lepiej użyć zwykłego BST. Jeżeli nie planujemy częstego usuwania i dodawania elementów, a jedynie wyszukiwanie, to lepszym wyborem jest drzewo AVL.


Tree Build Time
Build BST
Build AVL
0.0
0.1
0.2
0.3
0.4
2000
4000
6000
8000
10000


Find in Tree Time
Find in BST
Find in AVL
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2000
4000
6000
8000
10000


Removing from Tree Time
Find in BST
Find in AVL
0.20
0.15
0.10
0.05
0.00
2000
4000
6000
8000
10000