Lidia Sobońska 310903
Arkadiusz Kojtek 310746

AISDI

Sortowanie

Sprawozdanie

Oświadczamy że poniższa praca, będąca podstawą do ocenienia wyników pierwszych 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 do posortowania z terminala.
3. **timeit** wykorzystany do zmierzenia czasu jaki był potrzebny do posortowania przez poszczególne algorytmy.

Algorytmy sortujące

Poszczególne kody reprezentujące algorytmy sortowania zostały rozmieszczone w plikach .py gdzie odpowiednio:
bubsort.py - kod reprezentujący algorytm sortowania bąbelkowego
insertsort.py - kod reprezentujący algorytm sortowania przez wstawianie
mergesort.py - kod reprezentujący algorytm sortowania przez scalanie
quicksort.py - kod reprezentujący algorytm szybkiego sortowania

Kod sortowania bąbelkowego oraz szybkiego został napisany przez Lidię Sobońską, natomiast kod sortowania przez wstawianie i przez scalanie został napisany przez Arkadiusza Kojtek. Kod główny (plik **sort.py**) był pisany wspólnie na dzielonym ekranie z wykorzystaniem funkcji Live Share w Visual Studio Code.

Sposób uruchomienia programu

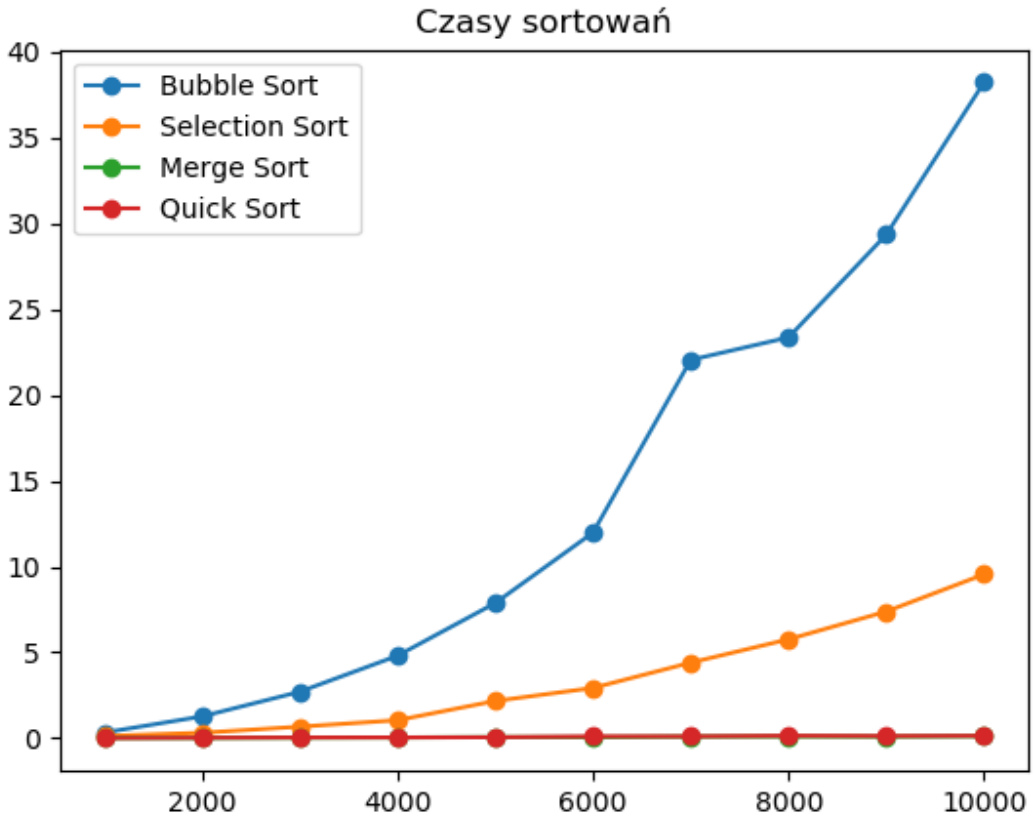
Plikiem obsługującym każdy z projektów jest **sort.py**, uruchomienie projektu polega na wywołaniu pliku **sort.py** z poziomu terminala z dopisanym argumentem reprezentującym plik zawierający dane do odczytu i posortowania. W przypadku, gdy nazwa pliku z danymi nie zostanie podana, domyślnie będzie brany plik ‘*pan-tadeusz.txt*’.

Przebieg działania programu

Projekt najpierw próbuje wczytać dane do posortowania z wybranego przez użytkownika pliku, po czym przechodzi do sortowania tych danych z użyciem poszczególnych algorytmów. Czas potrzebny na pełne sortowanie danych jest mierzony z użyciem funkcji **timeit()**. Informacje na temat długości pliku oraz czasu sortowania są na bieżąco zapisywane do tablic, które następnie wykorzystywane są do utworzenia wykresu z wykorzystaniem obiektu **pyplot**.

Pliki wynikowe

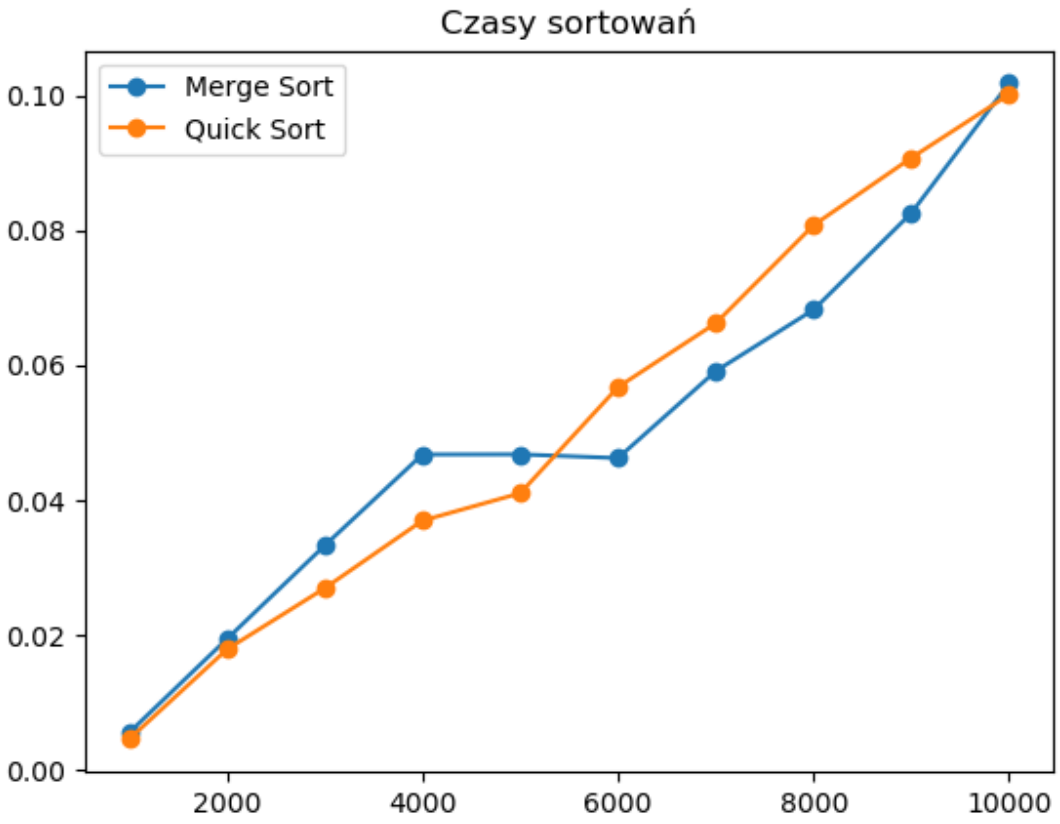
Jako plik wynikowy zostanie utworzony plik “wykres.png” zawierający graficzne przedstawienie przebiegów czasowych każdego z algorytmów.



Zależność czasu od liczby słów dla poszczególnych algorytmów przy jednym z testów				
Liczba słów	Bubble Sort (s)	Insertion Sort (s)	Merge Sort (s)	Quick Sort (s)
1000	0.3302144999997836	0.12919000000056258	0.01113939999959257	0.006897799999933341
2000	1.2696188999998412	0.3161920000002283	0.01531189999968774	0.027295500000036554
3000	2.7056711000004725	0.6736240999998699	0.03029400000013993	0.041873799999848416
4000	4.8045075	1.0497228000003815	0.04406550000021525	0.04612489999999525
5000	7.8734480000002804	2.171168700000635	0.06829770000058488	0.060099999999692955
6000	11.978049000000283	2.9291374999993423	0.05998769999951037	0.11035410000022239
7000	22.038490400000228	4.40474279999944	0.09993130000020756	0.10928119999971386
8000	23.34872180000002	5.762666999999965	0.10390379999989818	0.14513990000068588
9000	29.337267800000518	7.382504700000027	0.09802120000040304	0.1291320000000269
10000	38.212532000000465	9.550743799999509	0.12311469999986002	0.13983579999967333

Test pokazał wyraźnie dłuższy czas sortowania dla algorytmu bąbelkowego, który dla 10 tys. słów był prawie czterokrotnie dłuższy od drugiego z najwolniejszych algorytmów - sortowania przez wstawianie. Natomiast sortowanie szybkie i poprzez scalanie osiągnęły bardzo podobny czas z niewielką przewagą sortowania przez scalanie. Różnica jest jednak na tyle nieduża, że naszym zdaniem wręcz marginalna i wynik dla obu z tych algorytmów może dać różne wyniki dla innych danych przez co ciężko jest jednoznacznie ocenić, który z tych algorytmów jest lepszy.

Dla lepszego zobrazowania różnicy czasu pomiędzy sortowaniem szybkim i poprzez scalanie, na poniższym wykresie przedstawiono przebieg dla jedynie tych dwóch algorytmów.



Liczba słów	Merge Sort (s)	Quick Sort (s)
1000	0.005577	0.0046101
2000	0.0194489	0.0179114
3000	0.0333079	0.0269785
4000	0.0468086	0.0369992
5000	0.046826	0.0410651
6000	0.046273	0.0567918
7000	0.0591652	0.066265
8000	0.0682579	0.0807874
9000	0.0825938	0.0907768
10000	0.1018373	0.1001598

Anomalie wykresu, jak na przykład wyższy czas przy 4000 słów niż przy 6000 słów wynika najprawdopodobniej z ilości innych procesów w tle zachodzących w danym momencie. Przy tak niskich czasach wpływ tego może być widoczny, lecz są to różnice minimalne.