ASD - projekt nr. 1

autor: Jakub Żurawski

link do repo.: https://github.com/s23047-jz/algorytmy_project1

1. Krótki wstęp

Program został napisany w języku Python. Jego celem jest porównanie czasu działania czterech algorytmów sortujących. To przeprowadzenia testów pomiarowych posłużyłem się tablicami różnej wielkości. Tablice liczą od 300 tys. do 500 tys. elementów(w przedziale wartości liczb [1-300]. Tablice zostały wypełnione losowymi liczbami.

2. Wyniki

algorytmy sortujące	czas przy 300 tys. elem.	czas przy 500 tys. elem.
quicksort [s]	**~56**	**~162**
merge [s]	**~2**	**~3.8**
heapsort [s]	**~4**	**~7.6**
bubblesort [s]	**-**	**-**

Quicksort

Mimo swojej nazwy, czas jaki musiał otrzymać na wykonania zadania zajmuje znacznie dłużej od metod takich jak merge czy heap. Mimo wszystko czas jest w dobry.
Mówimy tutaj o kilkuset tysiącach elementów. Zwróćmy teraz uwagę na długość kodu oraz jego czytelność. Algorytm został napisany w dwóch funkcjach, co więcej nie zajął on dużej ilości linijek, a jego schemat jest czysty i łatwo się w nim połapać.

Mergesort

Spośród naszych rywalizujących ze sobą algorytmów sortujących, merge okazuje się zajmować pierwsze miejsce. Czas w jakim się zmieścił jest naprawdę mały. Wada jest jego zrozumiałość. Trzeba faktycznie kilka razy przeczytać kod aby go dobrze zrozumieć.

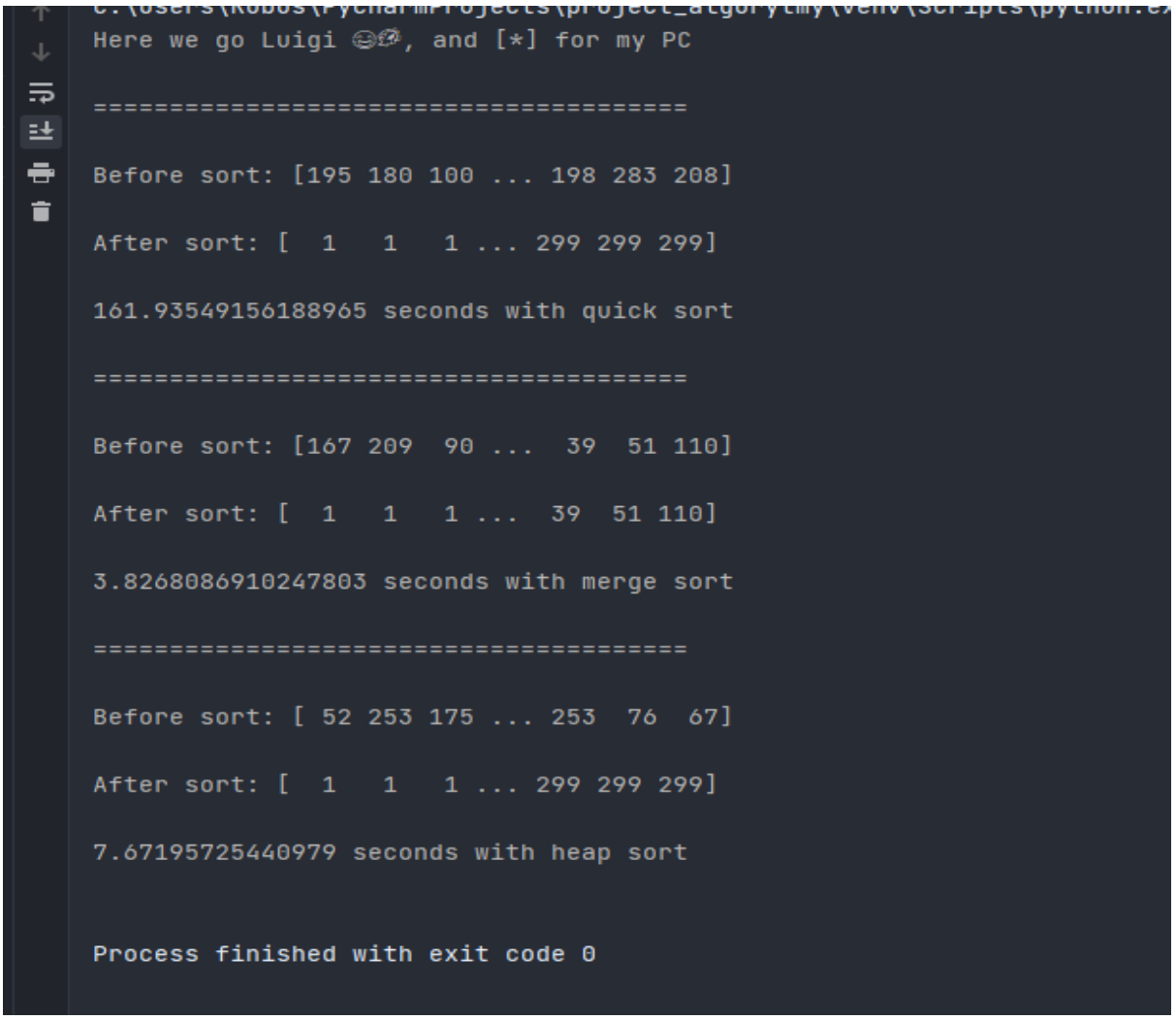
Heapsort

W zasadzie po opisaniu Mergesorta, wiele więcej o heapsort powiedzieć nie trzeba. Można powiedzieć, że jest tylko 2 razy wolniejszy od mergesorta. Plusem tego sortowania jest łatwiejszy do zrozumienia kod niż w przypadku merga.

Bubblesort

Jako ostatni, najwolniejszy bubblesort. Jest strasznie wolny. Czas nie został dokładnie zmierzony, ponieważ zajmowało to zbyt długo. Jego test polegał na sprawdzeniu poprawności sortowania dla tablicy długości równej 10. Przy testach dla 300 tys. poddałem się po dwóch godzinach. Ostatecznie dla dużych zbiorów danych nie polecam tej metody. Oczywiście, jeśli będziemy operować na małej liczbie danych ten algorytm może się dobrze sprawdzić ze względu na swój kod. Zajmuje raptem cztery linijki kodu. Jako, że zawsze staramy się jak najprościej i w jak najmniejszej ilości kodu rozwiązać problemu (nie zawsze się da) taki algorytm dla małych danych może się sprawdzić.

Konsola



Podsumowanie

Po przeprowadzonych testach śmiało można powiedzieć, że wygrywa mergesort. Jako, że jesteśmy ludźmi i pragniemy tego co lepsze, szybsze, wygodniejsze większość osób stanowczo użyje mergesorta. Musimy też zauważyć, że algorytmy są obecnie sprawdzane na nowszych maszynach, dla których posortowanie nawet 500 tys. elementów zajmuje cztery sekundy. W zależności od zamysłu programu, polecałbym dwa rozwiązania. Dla dużych danych merge, dla małych bubble. Mój wybór jest kierowany tym, że staramy się napisać kod tak, aby zajmował jak najmniej miejsca/linijek.