Algorytmy i struktury danych, Teleinformatyka, I rok

Raport z laboratorium nr: 4

Imię i nazwisko studenta: Mateusz Warzecha

nr indeksu: 410846

1. W pole poniżej wklej najważniejszy (według Ciebie) fragment kodu źródłowego z zajęć (maksymalnie 15 linii).

```
1.    while left<mid and right<end:
2.          if A[left]<A[right]:
3.                sorted_list.append(A[left])
4.                left+=1
5.          else:
6.                sorted_list.append(A[right])
7.                right+=1
8.    while left<=mid:
9.          sorted_list.append(A[left])
10.         left+=1
11.   while right <=end:
12.         sorted_list.append(A[right])
13.         right+=1
14.   for i in range(start,end+1):
15.         A[i]=sorted_list[i-start]
```

Uzasadnij swój wybór.

Wybrałem te 15 linijek ponieważ zawiera się w nich fragment funkcji merge który jest kluczowy do poprawnego działania algorytmu sortowania przez scalanie. Sortuje on elementy w podzielonych wcześniej fragmentach tablicy oraz scala je w odpowiedniej kolejności.

2. Podsumowanie:

```
Insertion sort time:50.897536754608154
Merge sort time:4.1506195068359375
Python sort() function time: 0.013011217117309 57
Total time: 56.139116525650024
```

Wyżej przedstawiony jest czas działania konkretnych algorytmów dla 1000 iteracji dla ciągów o długości 1000 i losowanych liczb z zakresu od 1 do 10 000. Wyraźnie widać że funkcja ***merge sort*** jest o wiele wydajniejsze niż ***insertion sort***. Mimo to funkcja ***.sort()*** bez wątpienia ma najkrótszy czas działania.