Prowadzący:
dr inż. Marcin Szlenk

13.11.2021r.

ALGORYTMY I STRUKTURY DANYCH

Laboratorium
Algorytmy sortowania

Wykonali:
Igor Nowak (SelectionSort, QuickSort)
Przemysław Wiszniewski (BubbleSort, MergeSort)

1. Cel projektu.

Celem projektu była implementacja kilku prostych algorytmów sortowania oraz zbadanie czasu ich wykonywania dla podanej tablicy wejściowej zawierającej słowa.

2. Środowisko programistyczne i platforma testowa.

Algorytmy oraz skrypt do wykonywania pomiarów zostały zaimplementowane w języku Python. Na platformę testową składał się komputer wyposażony w procesor AMD Ryzen 5 3600 taktowany zegarem 4.2 GHz operujący pod systemem Windows 10 Pro w wersji 21H1 oraz środowisko języka Python w wersji 3.9.4.

3. Pliki i funkcje.

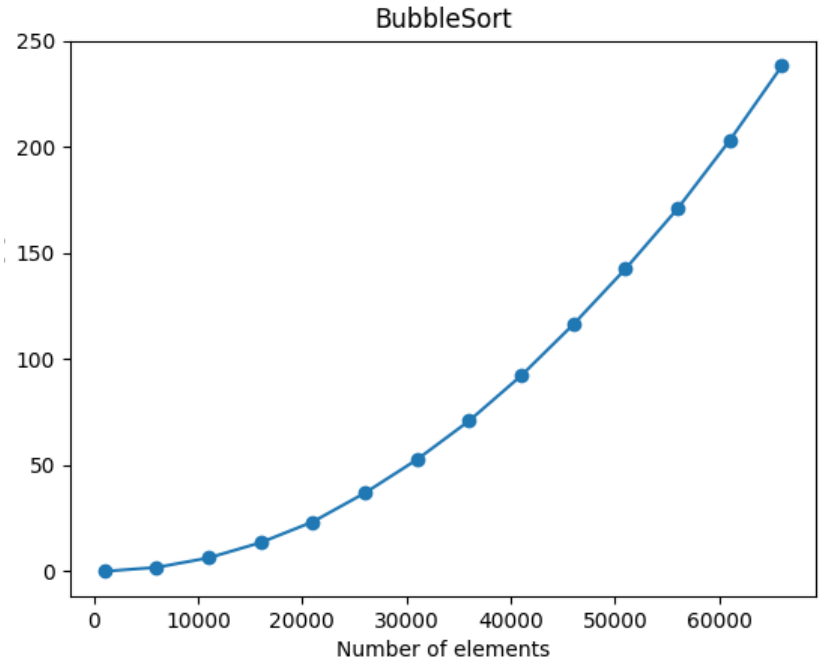
Skrypt testujący został podzielony na kilka plików składowych:

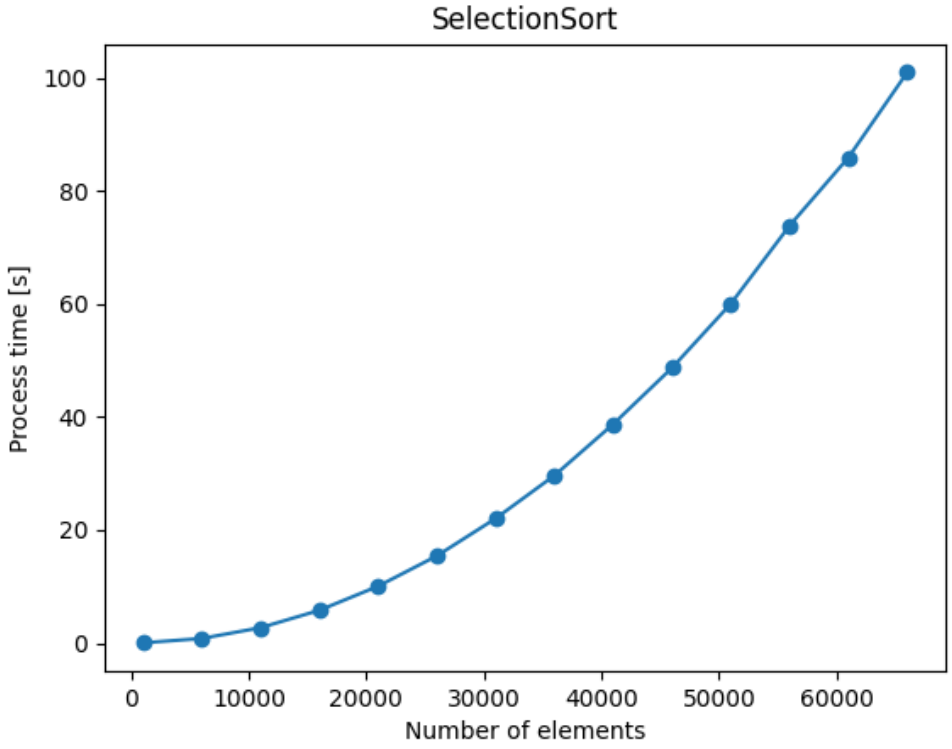
- **Main.py** – główny plik zawierający tylko jedną funkcję odpowiedzialną za wystartowanie pomiarów czasu dla zadanego pliku wejściowego oraz listy zawierającej ilość elementów do posortowania.
- **Utils.py** – plik zawierający klasę Utils, odpowiedzialną za obsługę wszystkich czynności niezbędnych do wykonania pomiarów, takich jak wczytanie danych, wystartowanie pomiarów, stworzenie wykresów i ich zapis do pliku oraz wyświetlanie używanych tablic słów.
- **BubbleSort.py** – plik zawierający klasę BubbleSort implementującą algorytm sortowania bąbelkowego.
- **MergeSort.py** – plik zawierający klasę MergeSort implementującą algorytm sortowania przez scalanie.
- **QuickSort.py** – plik zawierający klasę QuickSort implementującą algorytm sortowania szybkiego.
- **SelectionSort.py** – plik zawierający klasę BubbleSort implementującą algorytm sortowania przez wybieranie.

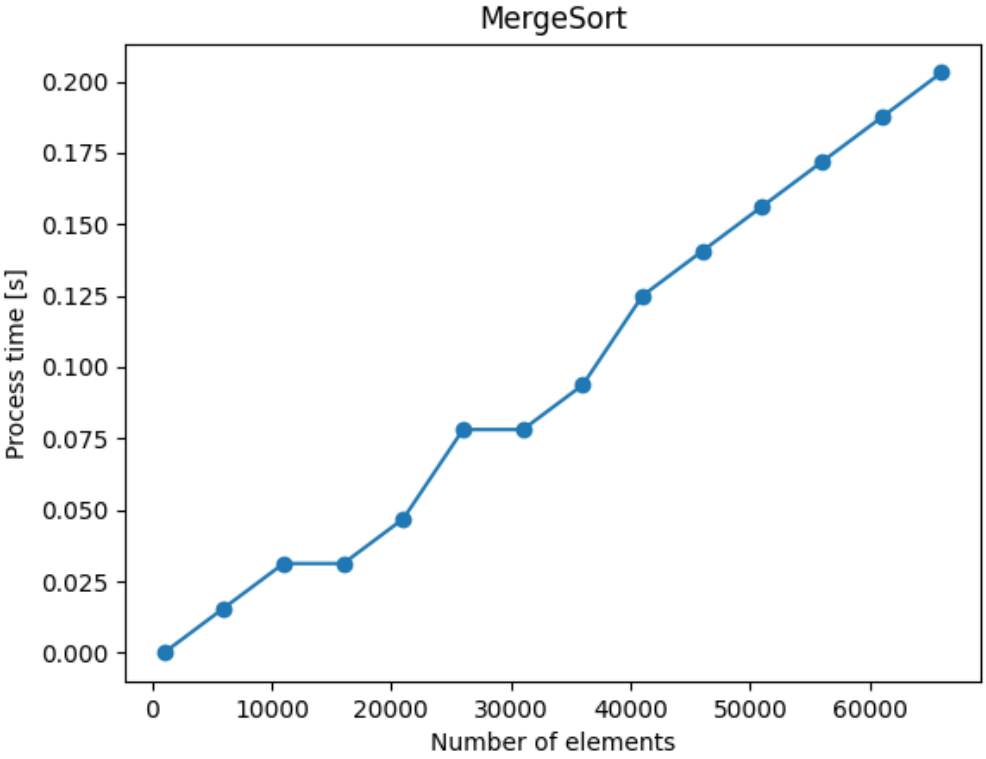
W celu uruchomienia pomiarów należy w konsoli użyć polecenia ***python Main.py***. Aby zmienić wejściowy plik zawierający słowa oraz liczbę elementów należy w pliku **Main.py** zmodyfikować zmienne ***input_file_name*** oraz ***number_of_elements_list***.

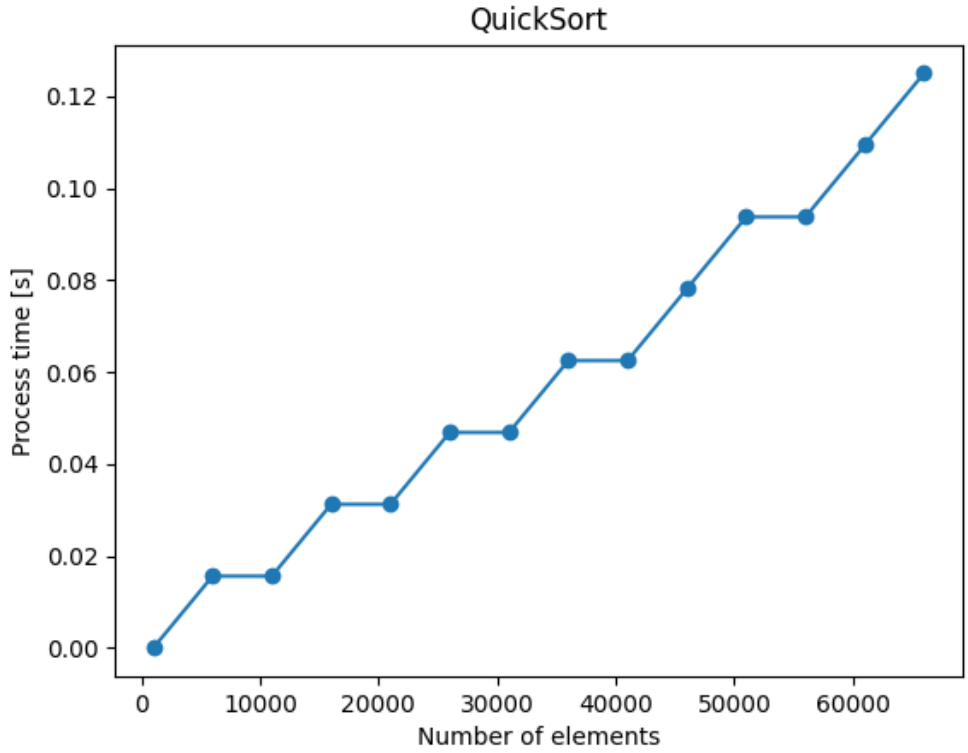
4. Wyniki pomiarów.

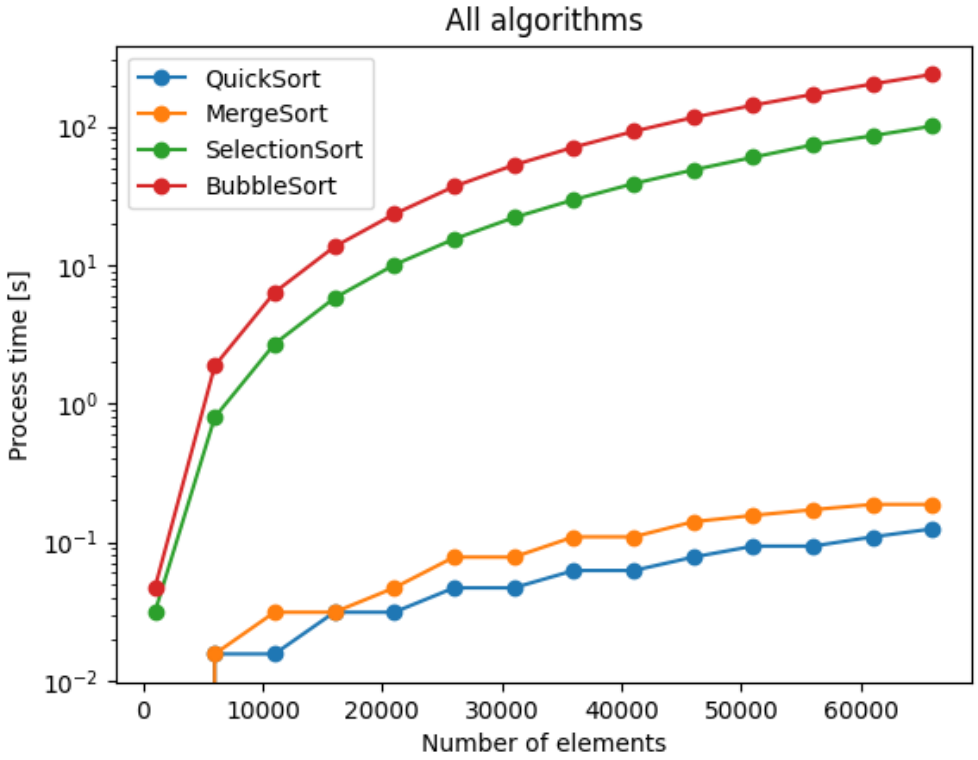
Pomiary zostały wykonane dla liczby elementów z przedziału 1000 – 66000 z krokiem 5000.




SelectionSort
Process time [s]
0
20
40
60
80
100
0
10000
20000
30000
40000
50000
60000
Number of elements


MergeSort
Process time [s]
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0
10000
20000
30000
40000
50000
60000
Number of elements


QuickSort
Process time [s]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
10000
20000
30000
40000
50000
60000
Number of elements


All algorithms
QuickSort
MergeSort
SelectionSort
BubbleSort
Process time [s]
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
0
10000
20000
30000
40000
50000
60000
Number of elements

5. Podsumowanie.

Algorytmy takie jak Bubble Sort i Selection Sort charakteryzują się złożonością $O(n^2)$ co również potwierdzają pomiary. Czas ich wykonania jest znacznie dłuższy od pozostałych, w szczególności algorytmu sortowania przez wybieranie, gdzie czas sortowania 66 tys. elementów zajmuje ponad 100 sekund.

Wykonanie algorytmów Merge Sort i Quick Sort jest znacznie szybsze. Ich złożoność wynosi $O(n*\log n)$. W algorytmie Quick Sort został wykorzystany zrównoważony podział, tak więc jego szybkość jest bardzo podobna do algorytmu Merge Sort. Posortowanie 66 tyś elementów algorytmem szybkiego sortowania zajmuje około 0.1 sekundy, a algorytmem sortowania przez scalanie około 0.2 sekundy.

Podsumowując, najszybszy z badanych algorytmów okazał się być algorytm szybkiego sortowania. Tuż za nim znalazł się algorytm sortowania przez scalanie, a następnie algorytm sortowania przez wybieranie. Najwolniejszym okazał się być algorytm sortowania bąbelkowego.