

ALGORYTMY I STRUKTURY DANYCH

Laboratorium Algorytmy sortowania

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.

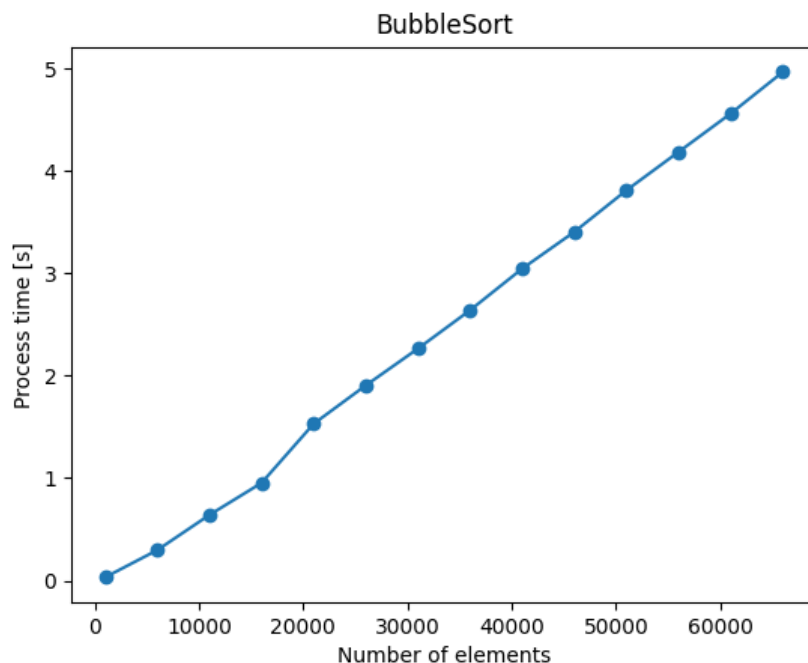
Skrypt testujący został podzielony na kilka plików składowych:

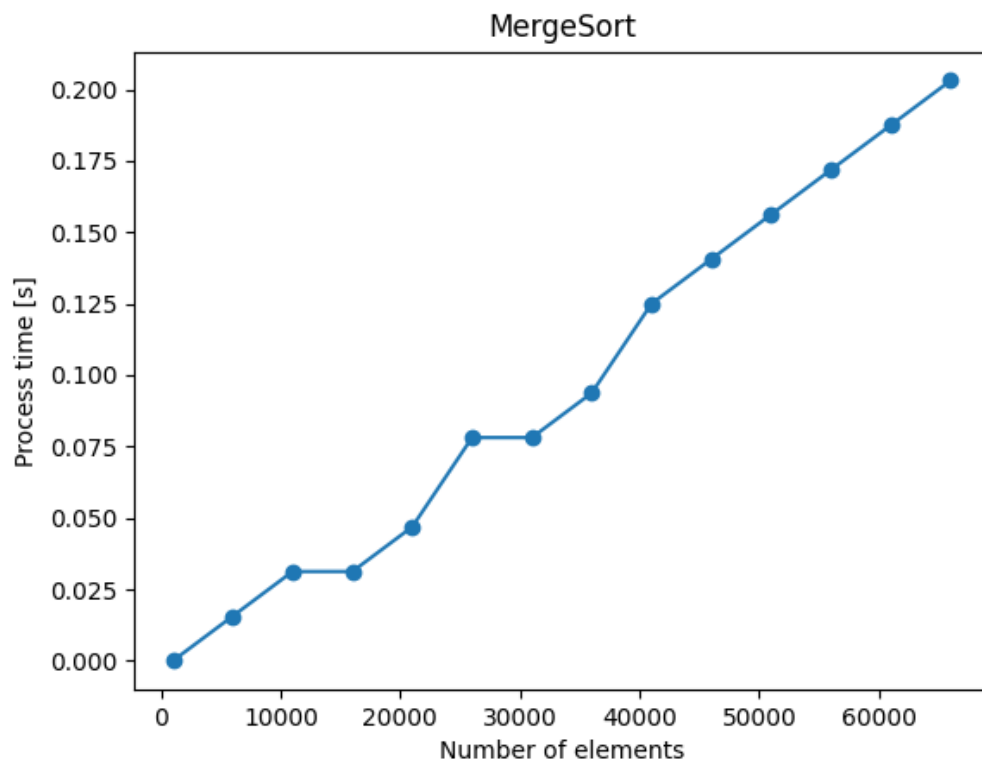
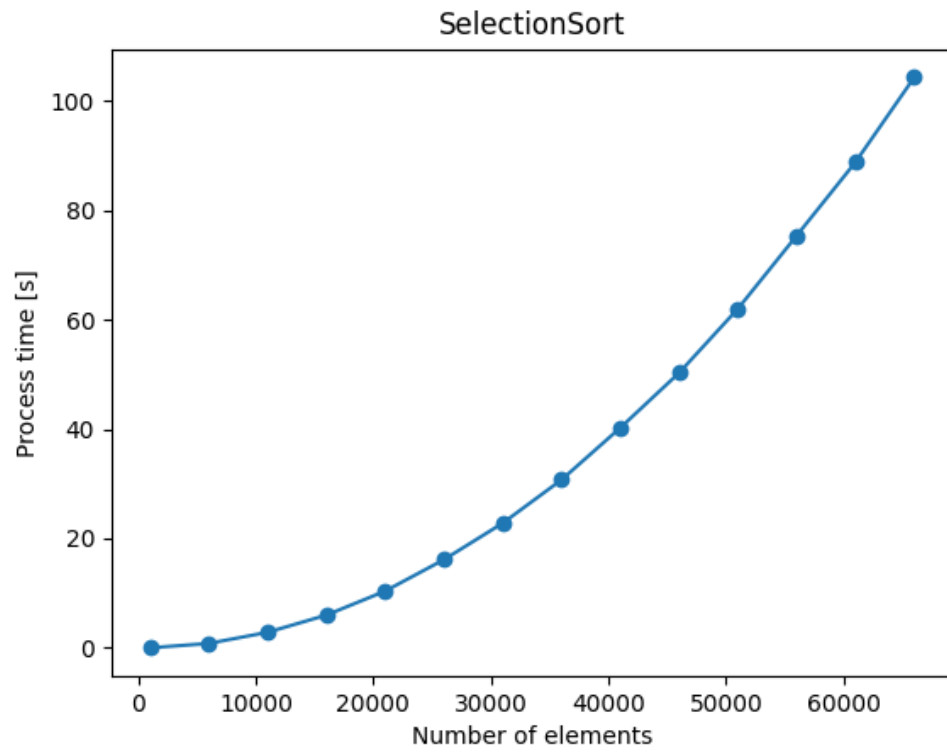
- **Main.py** – główny plik zawierający tylko jedną funkcję odpowiedzialną za wystartowanie pomiarów czasu dla zadanego pliku wejściowego oraz listy zawierającej ilość elementów do posortowania.
- **Utils.py** – plik zawierający klasę Utils, odpowiedzialną za obsługę wszystkich czynności niezbędnych 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.

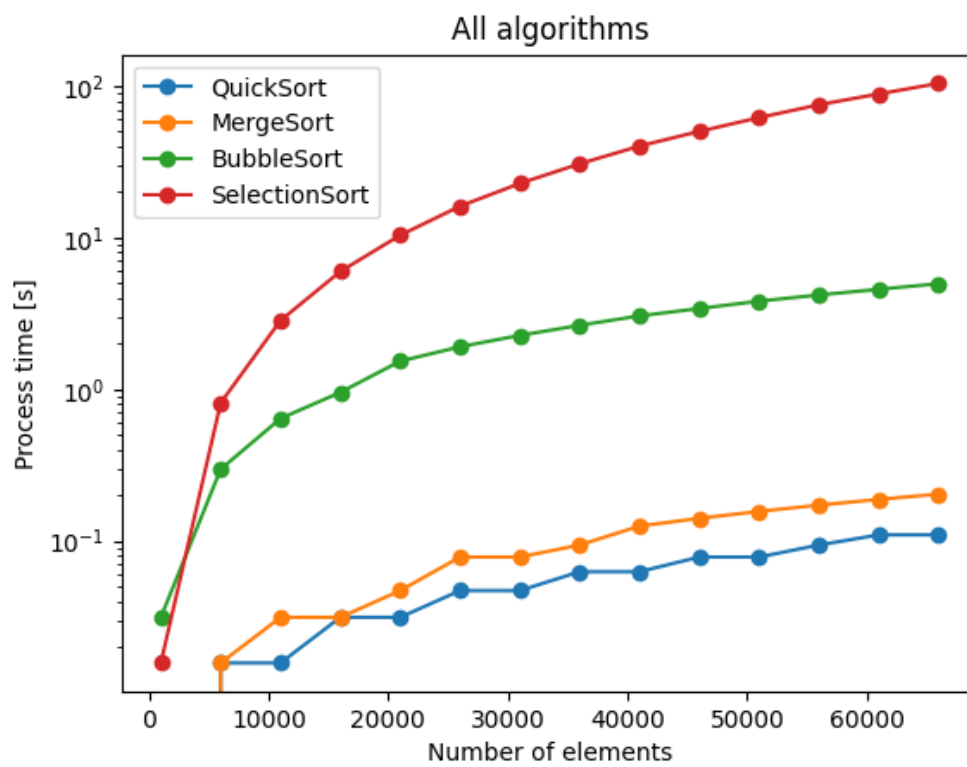
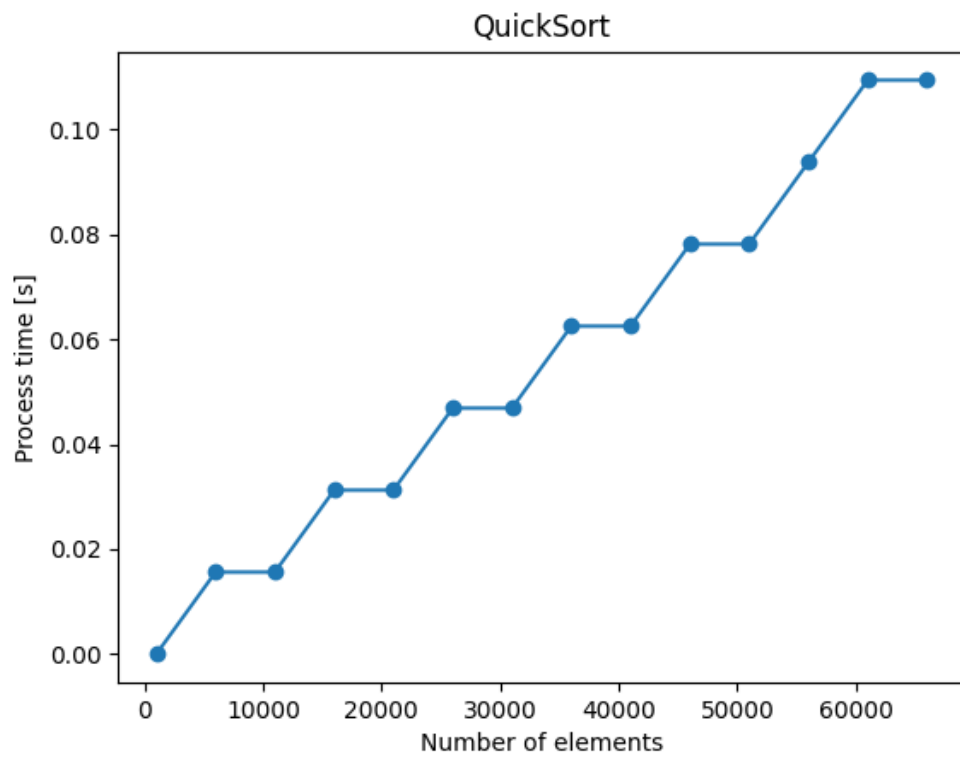
W celu uruchomienia pomiarów należy w konsoli użyć polecenia **python Main.py**. Aby zmienić wejściowy plik zawierający słowa oraz liczbę elementów należy w pliku **Main.py** zmodyfikować zmienne **input_file_name** oraz **number_of_elements_list**.

4. Wyniki pomiarów.

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







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 $O(n \cdot \log n)$. W algorytmie Quick Sort został wykorzystany zrównoważony podział, tak więc jego szybkość jest bardzo podobna do algorytmu Merge Sort. Posortowanie 66 tys 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 bąbelkowego. Najwolniejszym okazał się być algorytm sortowania przez wybieranie.