

Politechnika Warszawska
Algorytmy i struktury danych

Laboratorium 2
Projekt "Sortowanie"
Informatyka – Inteligentne systemy

Paweł Sarnacki 305290

Piotr Niedziałek 304474

Prowadzący: dr inż. Łukasz Skonieczny

Warszawa 2023

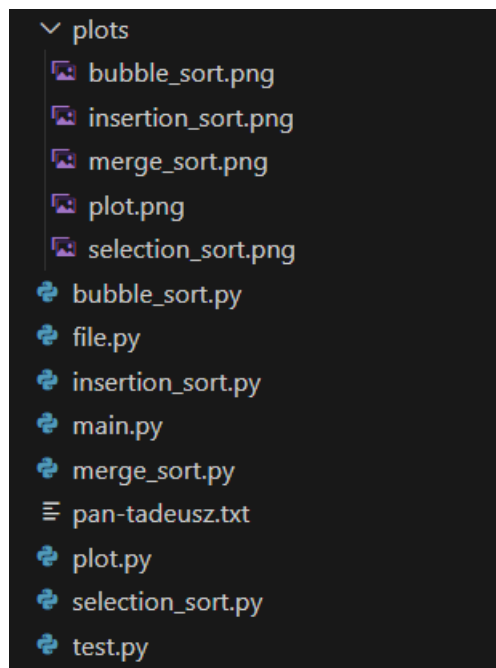
1. Wstęp

Program został napisany w języku python z wykorzystaniem aplikacji Visual Studio Code. Program tworzy posortowane tablice o rozmiarze n , składające się z n wyrazów z wybranego pliku tekstowego. Dodatkowo tworzone są wykresy czasu sortowania w zależności od ilości elementów n dla każdego algorytmu sortującego. Program był pisany w wersji python 3.9.13, a wykorzystane biblioteki to: time, matplotlib.pyplot, unicode i re.

2. Struktura projektu

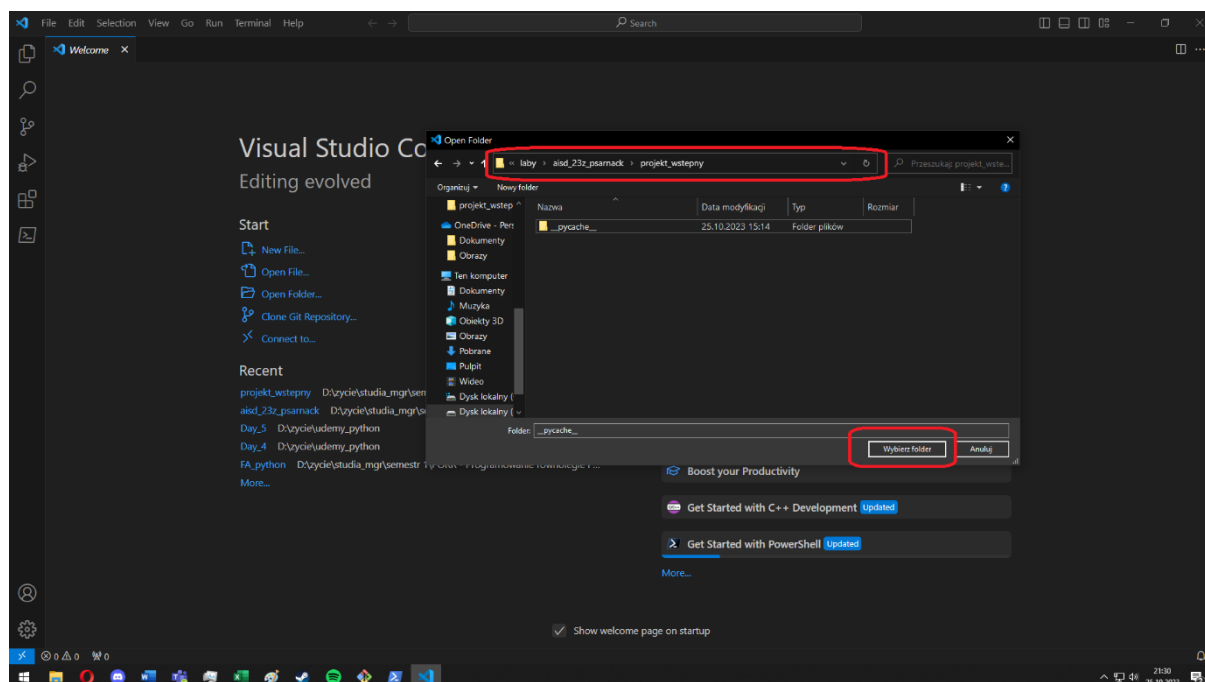
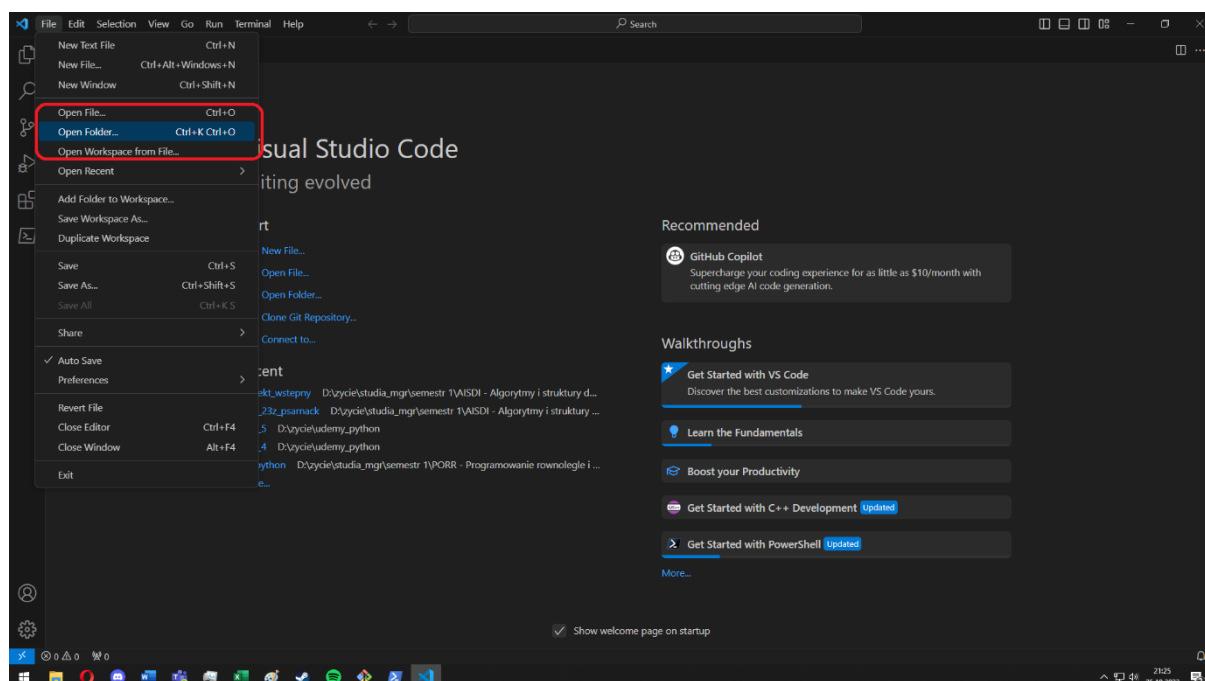
Projekt składa się z:

- Pliku main.py, który zawiera główny program obsługujący wszystkie algorytmy.
- Pliku insertion_sort.py, który zawiera implementację algorytmu sortowania przez wstawianie.
- Pliku bubble_sort.py, który zawiera implementację algorytmu sortowania bąbelkowego.
- Pliku merge_sort.py, który zawiera implementację algorytmu sortowania przez scalanie.
- Pliku selection_sort.py, który zawiera implementację algorytmu sortowania przez wybieranie.
- Pliku file.py, który zawiera implementację wczytywania n wyrazów i tworzenia listy.
- Pliku test.py, który zawiera implementację sprawdzania czy algorytmy posortowały wyrazy w ten sam sposób co wbudowana funkcja sorted().
- Pliku pan-tadeusz.txt, który zawiera w sobie tekst do sortowania.
- Katalogu plots, który zawiera wygenerowane wykresy.

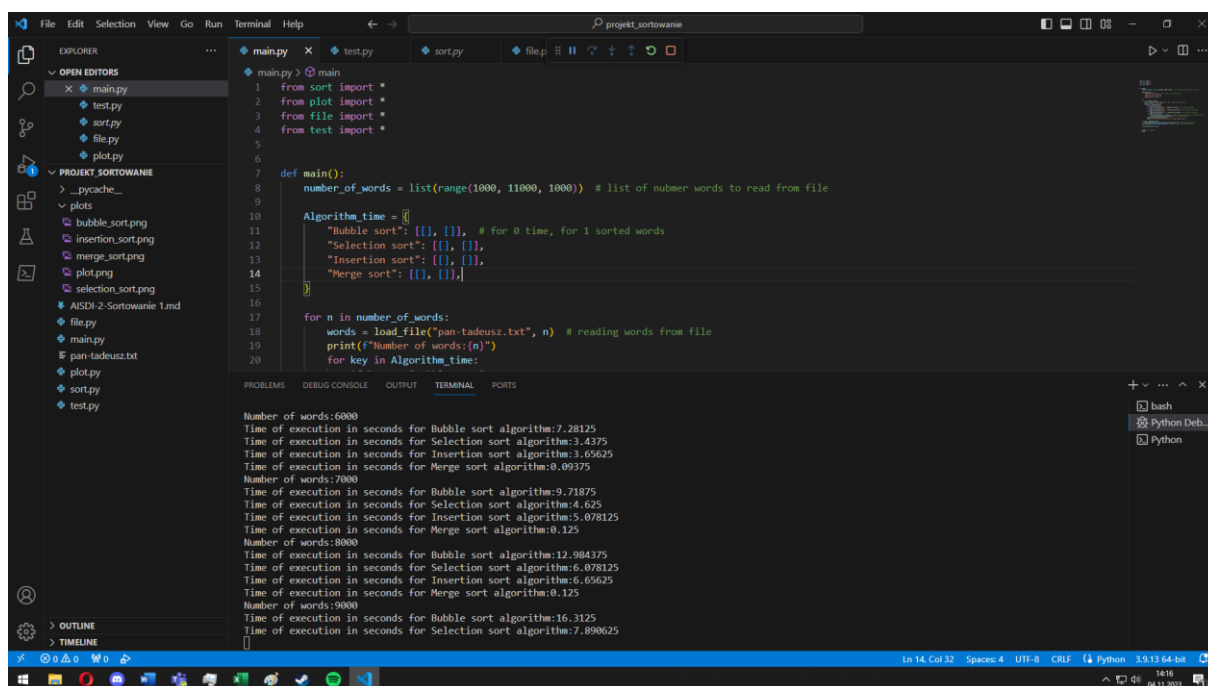


3. Uruchomienie projektu z środowiska Visual Studio Code

3.1. Otworzenie folderu, wybór ścieżki i kliknięcie wybierz folder



3.2. Wybranie terminalu bash, wpisanie „py main.py”, lub kliknięcie F5, na dole zostanie wyświetlony wynik

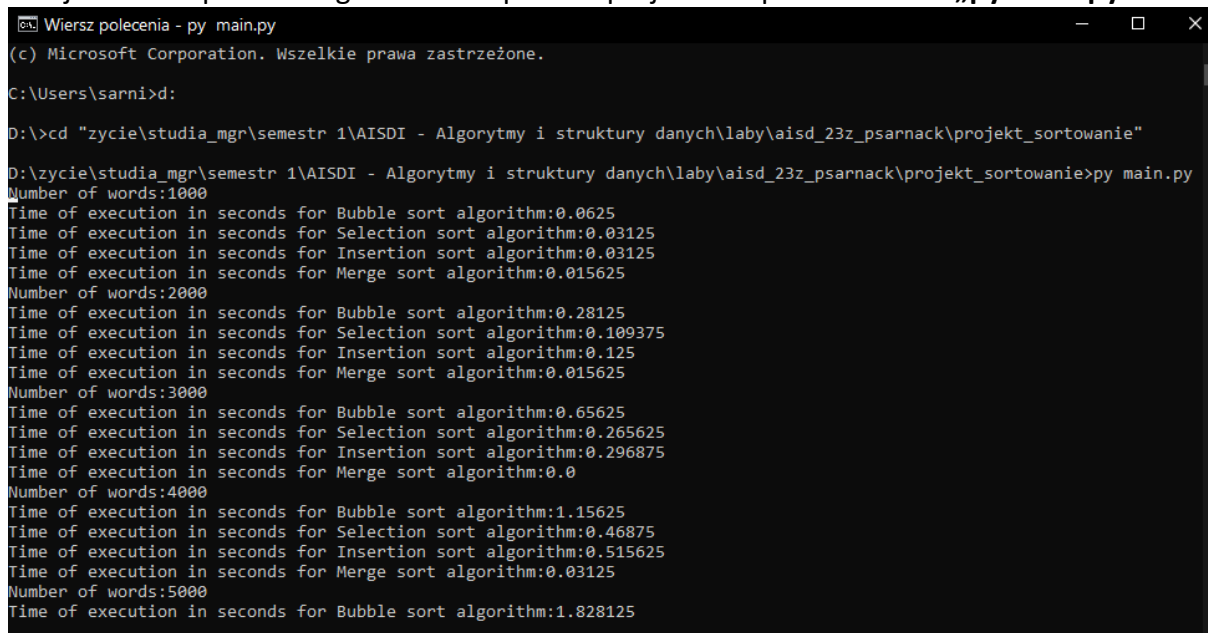


```
1 from sort import *
2 from plot import *
3 from file import *
4 from test import *
5
6
7
8 def main():
9     number_of_words = list(range(1000, 11000, 1000)) # list of number words to read from file
10
11     Algorithm_time = {}
12     "Bubble sort": ([], []), # for 0 time, for 1 sorted words
13     "Selection sort": ([], []),
14     "Insertion sort": ([], []),
15     "Merge sort": ([], []),
16
17     for n in number_of_words:
18         words = load_file("pan-tadeusz.txt", n) # reading words from file
19         print(f"Number of words:{n}")
20         for key in Algorithm_time:
```

```
Number of words:6000
Time of execution in seconds for Bubble sort algorithm:7.28125
Time of execution in seconds for Selection sort algorithm:3.4375
Time of execution in seconds for Insertion sort algorithm:3.65625
Time of execution in seconds for Merge sort algorithm:0.09375
Number of words:7000
Time of execution in seconds for Bubble sort algorithm:9.71875
Time of execution in seconds for Selection sort algorithm:4.625
Time of execution in seconds for Insertion sort algorithm:5.078125
Time of execution in seconds for Merge sort algorithm:0.125
Number of words:8000
Time of execution in seconds for Bubble sort algorithm:12.984375
Time of execution in seconds for Selection sort algorithm:6.078125
Time of execution in seconds for Insertion sort algorithm:6.65625
Time of execution in seconds for Merge sort algorithm:0.125
Number of words:9000
Time of execution in seconds for Bubble sort algorithm:16.3125
Time of execution in seconds for Selection sort algorithm:7.890625
```

4. Uruchomienie projektu bez środowiska Visual Studio Code

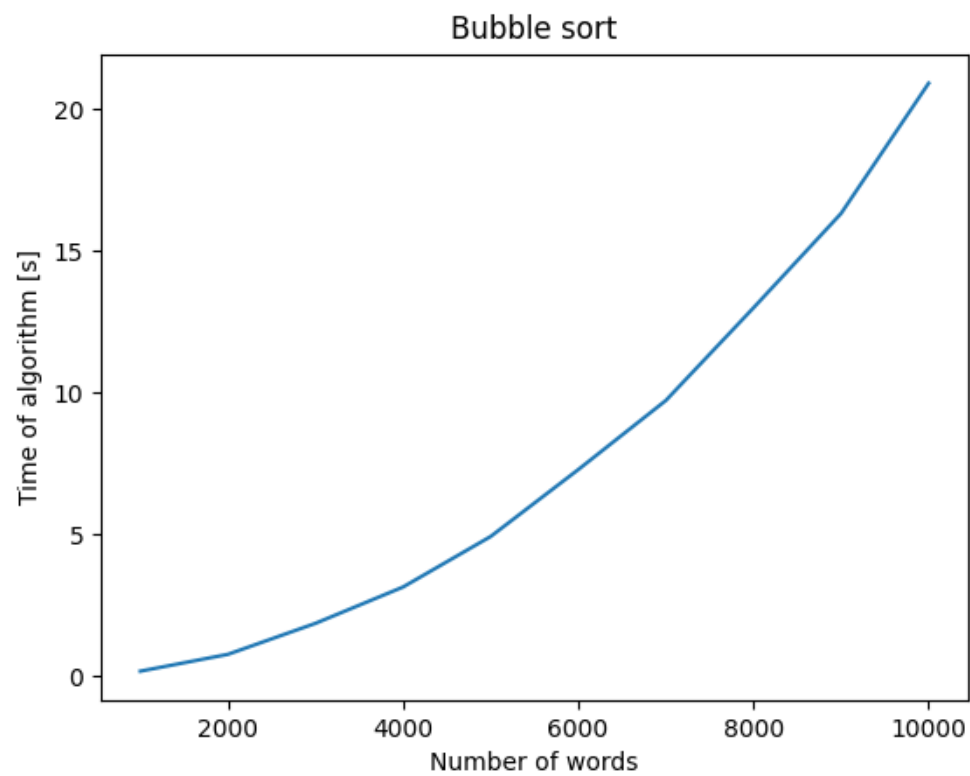
Przejdźcie do odpowiedniego folderu z plikami projektu i wpisanie w cmd „py main.py”



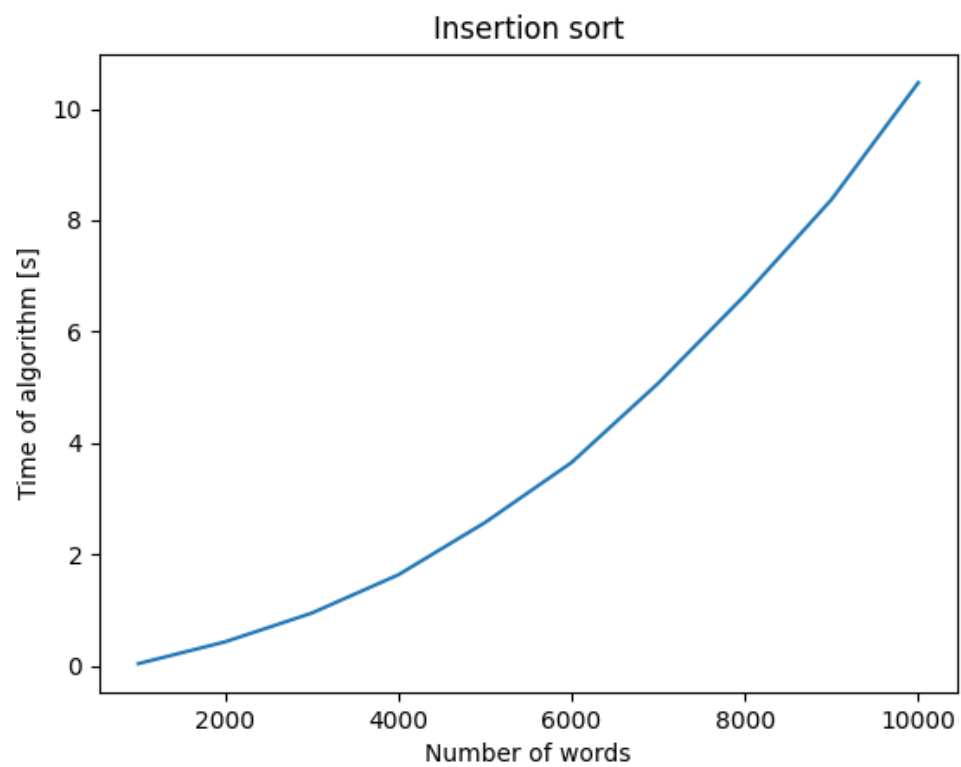
```
Wiersz polecenia - py main.py
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.
C:\Users\sarni>d:
D:\>cd "zycie\studia_mgr\semestr 1\AISDI - Algorytmy i struktury danych\laby\aisd_23z_psarnack\projekt_sortowanie"
D:\zycie\studia_mgr\semestr 1\AISDI - Algorytmy i struktury danych\laby\aisd_23z_psarnack\projekt_sortowanie>py main.py
Number of words:1000
Time of execution in seconds for Bubble sort algorithm:0.0625
Time of execution in seconds for Selection sort algorithm:0.03125
Time of execution in seconds for Insertion sort algorithm:0.03125
Time of execution in seconds for Merge sort algorithm:0.015625
Number of words:2000
Time of execution in seconds for Bubble sort algorithm:0.28125
Time of execution in seconds for Selection sort algorithm:0.109375
Time of execution in seconds for Insertion sort algorithm:0.125
Time of execution in seconds for Merge sort algorithm:0.015625
Number of words:3000
Time of execution in seconds for Bubble sort algorithm:0.65625
Time of execution in seconds for Selection sort algorithm:0.265625
Time of execution in seconds for Insertion sort algorithm:0.296875
Time of execution in seconds for Merge sort algorithm:0.0
Number of words:4000
Time of execution in seconds for Bubble sort algorithm:1.15625
Time of execution in seconds for Selection sort algorithm:0.46875
Time of execution in seconds for Insertion sort algorithm:0.515625
Time of execution in seconds for Merge sort algorithm:0.03125
Number of words:5000
Time of execution in seconds for Bubble sort algorithm:1.828125
```

5. Otrzymane wyniki

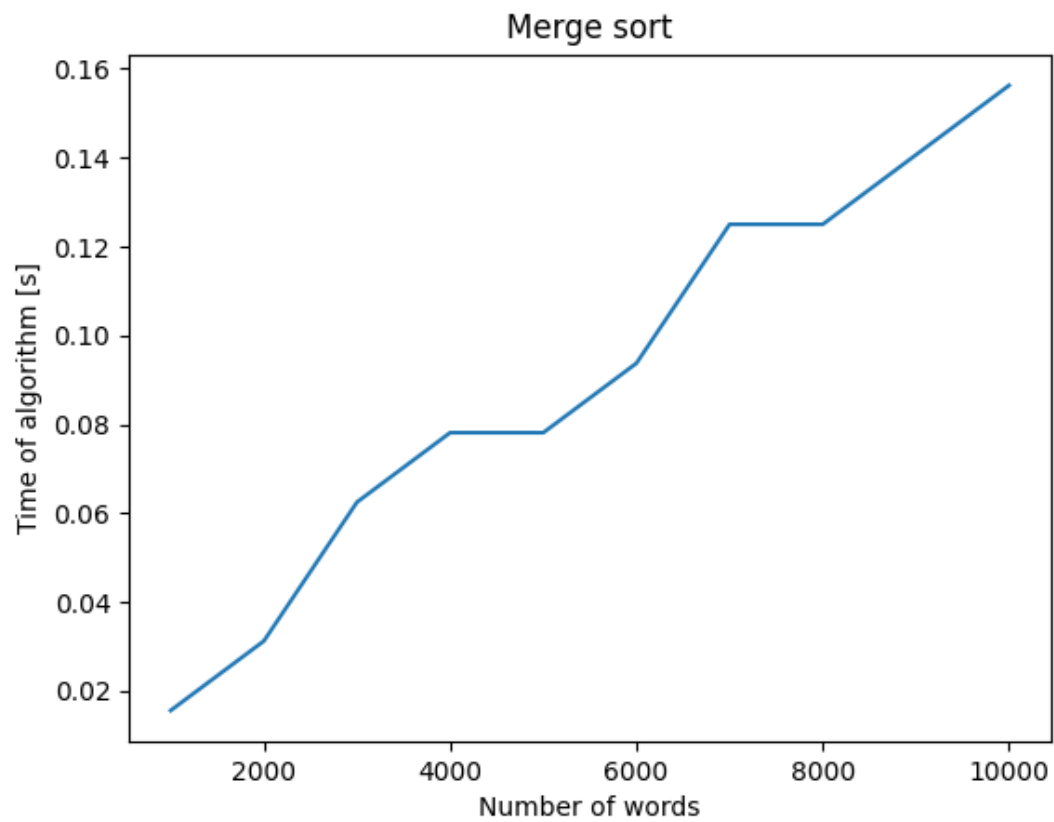
5.1. Bubble sort



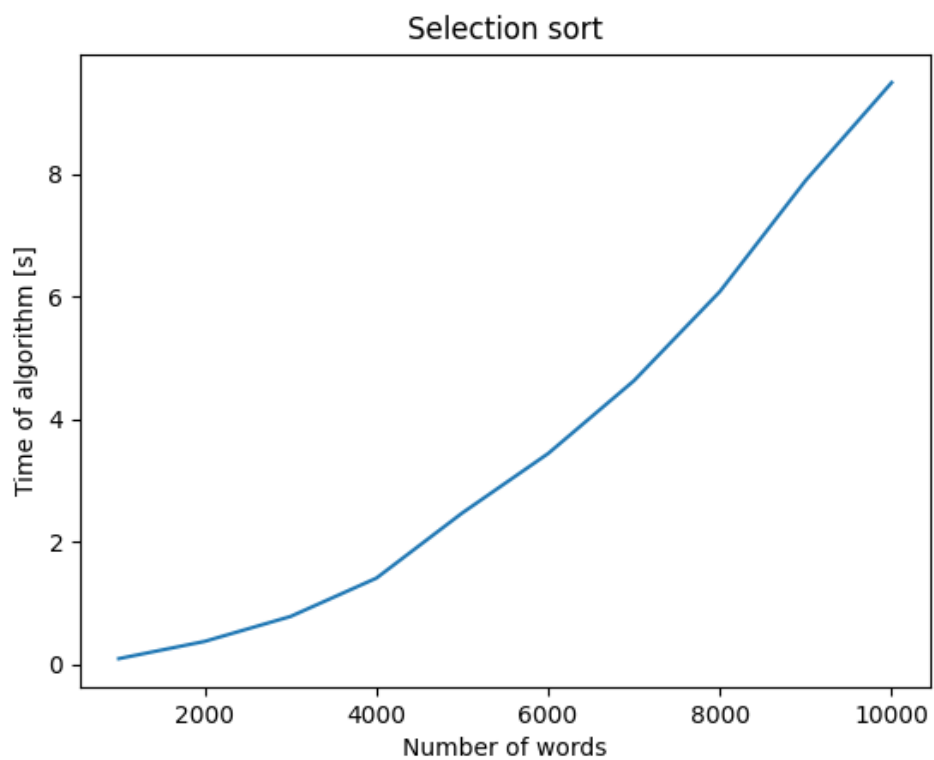
5.2. Insertion sort



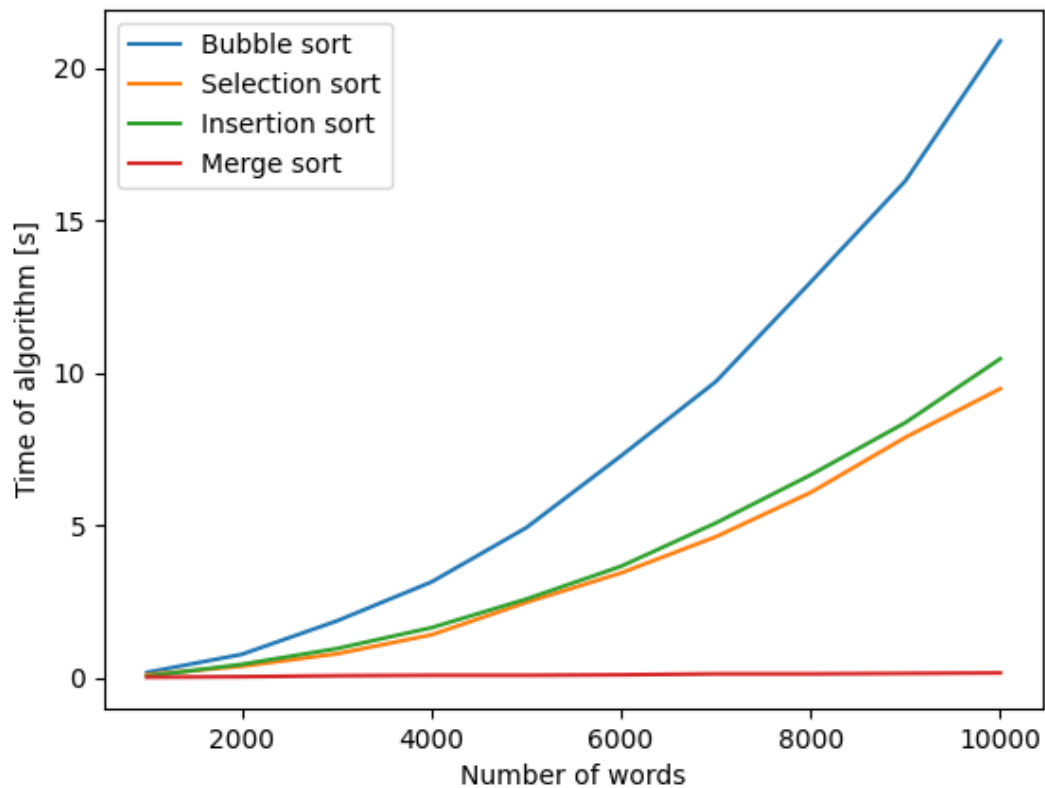
5.3. Merge sort



5.4. Selection sort



5.5. Wszystkie



5.6. Sprawdzenie

```
Number of words:8000
Time of execution in seconds for Bubble sort algorithm:12.984375
Time of execution in seconds for Selection sort algorithm:6.078125
Time of execution in seconds for Insertion sort algorithm:6.65625
Time of execution in seconds for Merge sort algorithm:0.125
Number of words:9000
Time of execution in seconds for Bubble sort algorithm:16.3125
Time of execution in seconds for Selection sort algorithm:7.890625
Time of execution in seconds for Insertion sort algorithm:8.375
Time of execution in seconds for Merge sort algorithm:0.140625
Number of words:10000
Time of execution in seconds for Bubble sort algorithm:20.90625
Time of execution in seconds for Selection sort algorithm:9.484375
Time of execution in seconds for Insertion sort algorithm:10.46875
Time of execution in seconds for Merge sort algorithm:0.15625
All algorithms sorted words the same way as a built in function sorted(), therefore we can assume that they might work correctly
```

6. Wnioski

- Najlepsze wyniki uzyskał algorytm sortowania przez scalanie (merge sort) Takiego rezultatu można było się spodziewać, ponieważ jego złożoność w najgorszym wypadku wynosi $O(n \log(n))$, natomiast pozostałych wynosi $O(n^2)$.
- Najgorsze wyniki uzyskał algorytm bąbelkowy.
- Wyniki dla algorytmu przez wstawianie i wybieranie były zbliżone do siebie.
- Zaimplementowane algorytmy działały poprawnie, to znaczy zwracały taką samą listę słów co wbudowana funkcja sorted().