AISD Sprawozdanie LAB 2 - Sortowanie

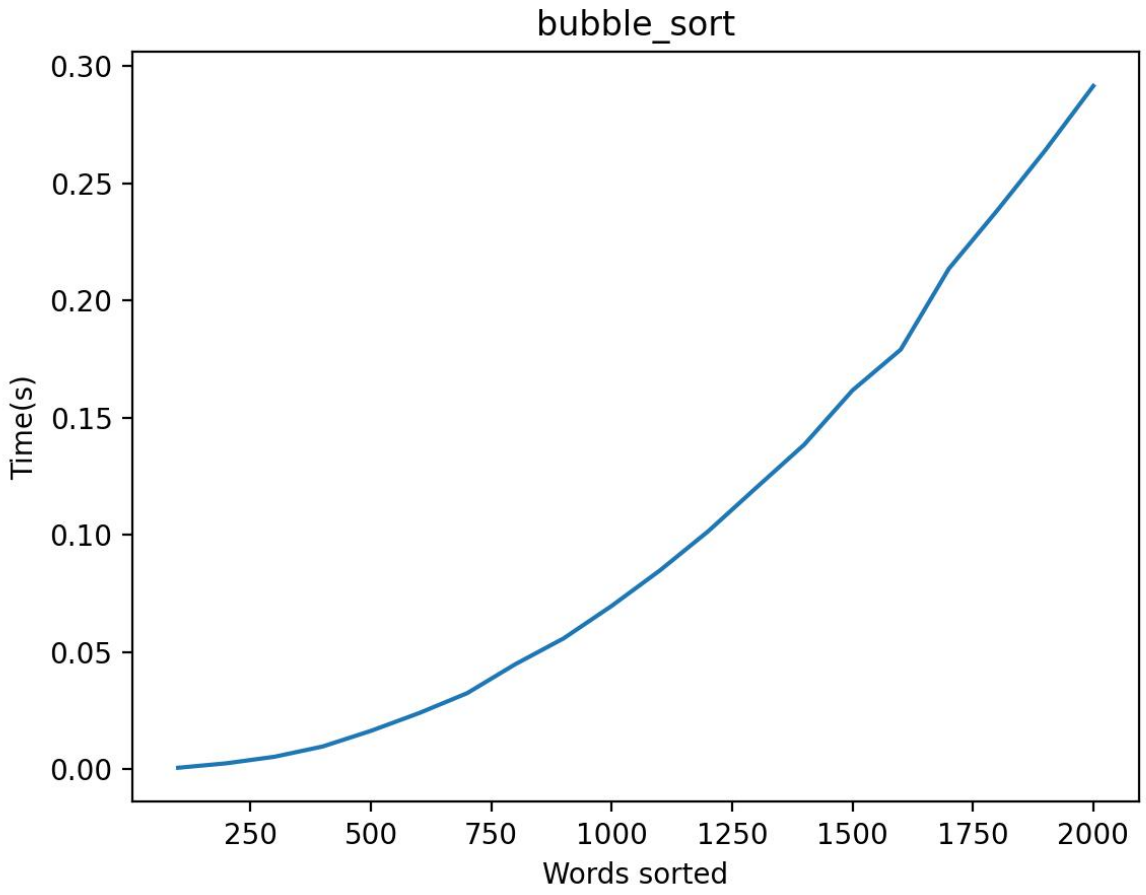
Zadanie

Naszym zadaniem było zaimplementowanie oraz porównanie czterech algorytmów sortowania:

- Bubble sort
- Selection sort
- Quicksort
- Merge sort

Zadanie zostało wykonane w pythonie, do sporządzenia wykresów użyto biblioteki matplotlib. Wszystkie algorytmy zostały wywołane na wcześniej przetworzonych danych z pliku `pan_tadeusz.txt` (pominięto wszystkie wyrazy zawierające nie-litery, oraz zamieniono wszystko na małe litery)

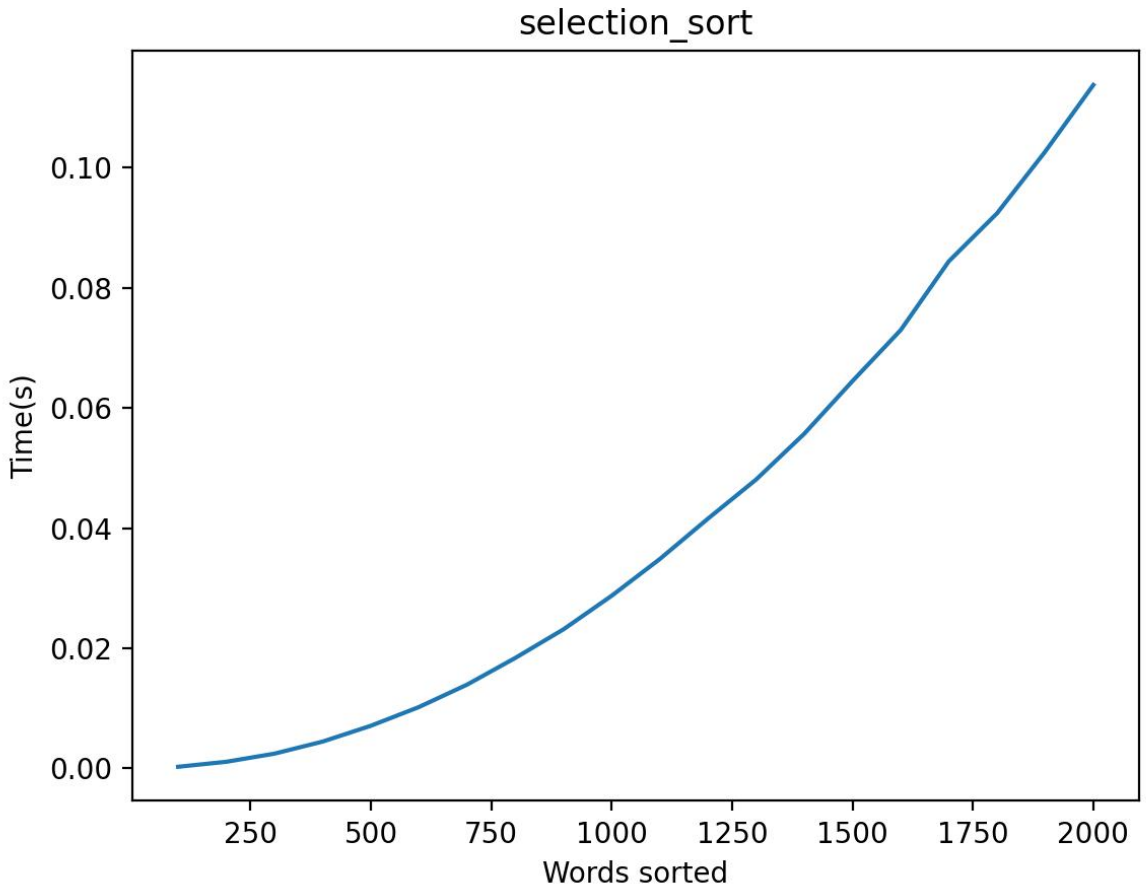
Bubble sort



Algorytm dobrze sprawdza się dla małego zestawu danych, lecz szybko ucieka do bardzo wysokich czasów wykonania wraz z rosnącą ilością słów.

Złożoność czasowa O(n^2)

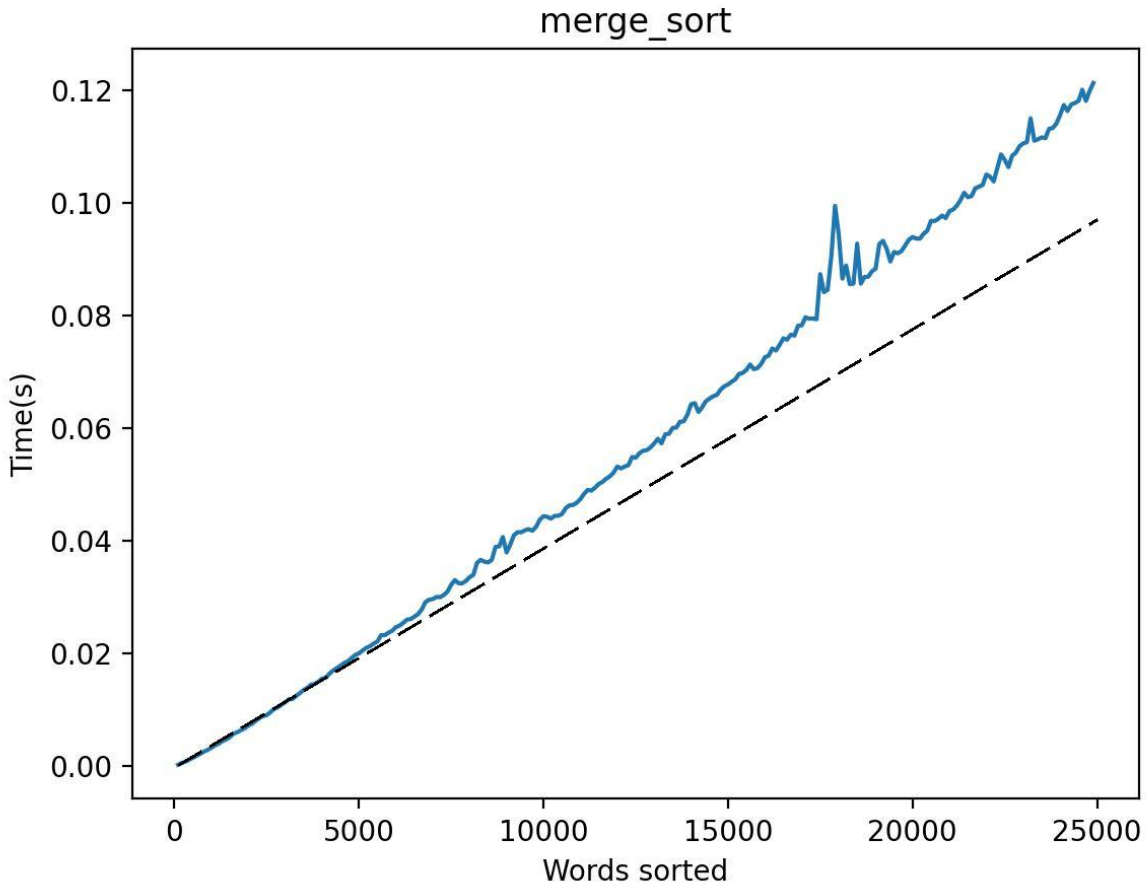
Selection sort



Podobnie do poprzedniego algorytmu zachodzi tendencja do szybkiego wzrostu czasu, jednak pozostaje kilkukrotnie szybszy. Wynika to z tego, że wykonywanych jest mniej kosztownych operacji zamiany miejscami elementów.

Złożoność czasowa O(n^2)

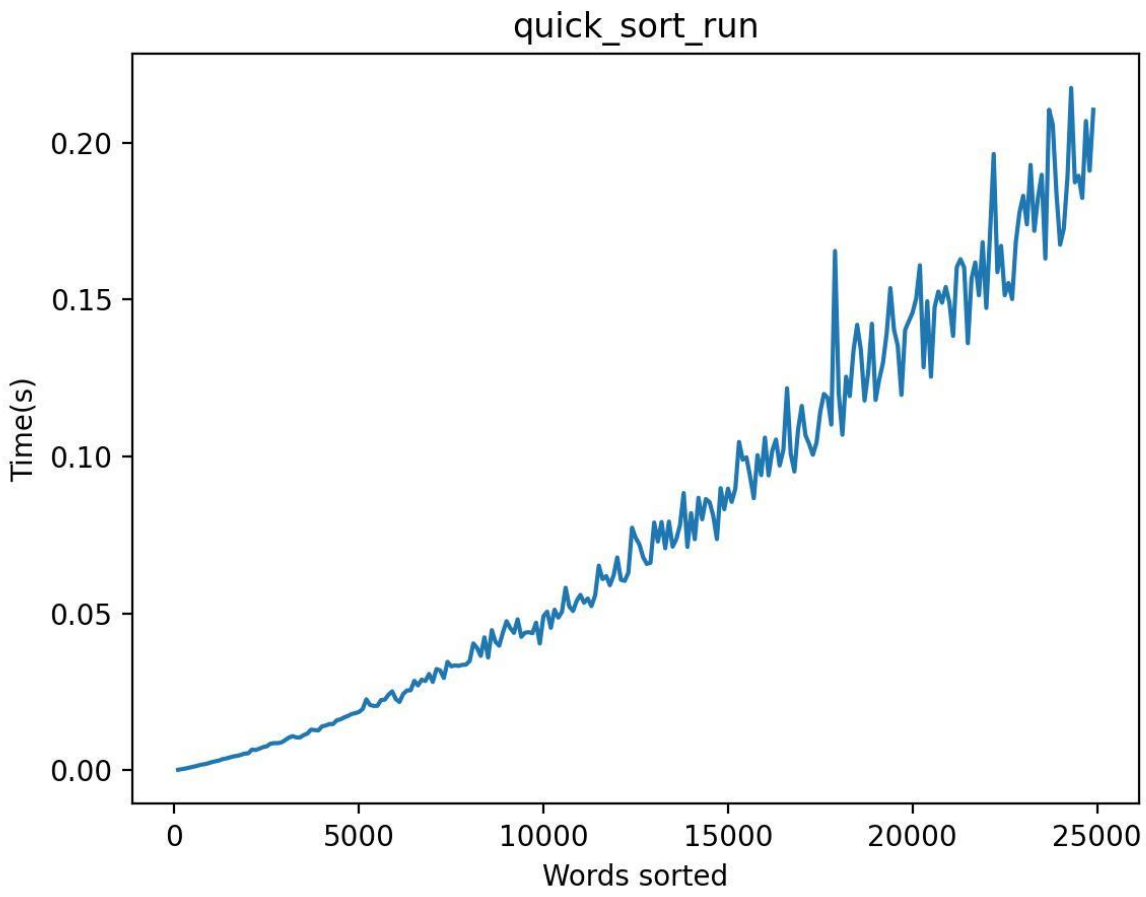
Merge sort



Algorytm stosujący zasadę „dziel i zwyciężaj”, osiąga znacznie lepsze rezultaty niż poprzednie prostrze algorytmy. Na początku wygląda jak funkcja liniowa, lecz powoli odchyla się ku górze (dla porównania zaznaczono na czarno)

Złożoność czasowa: nlog(n)

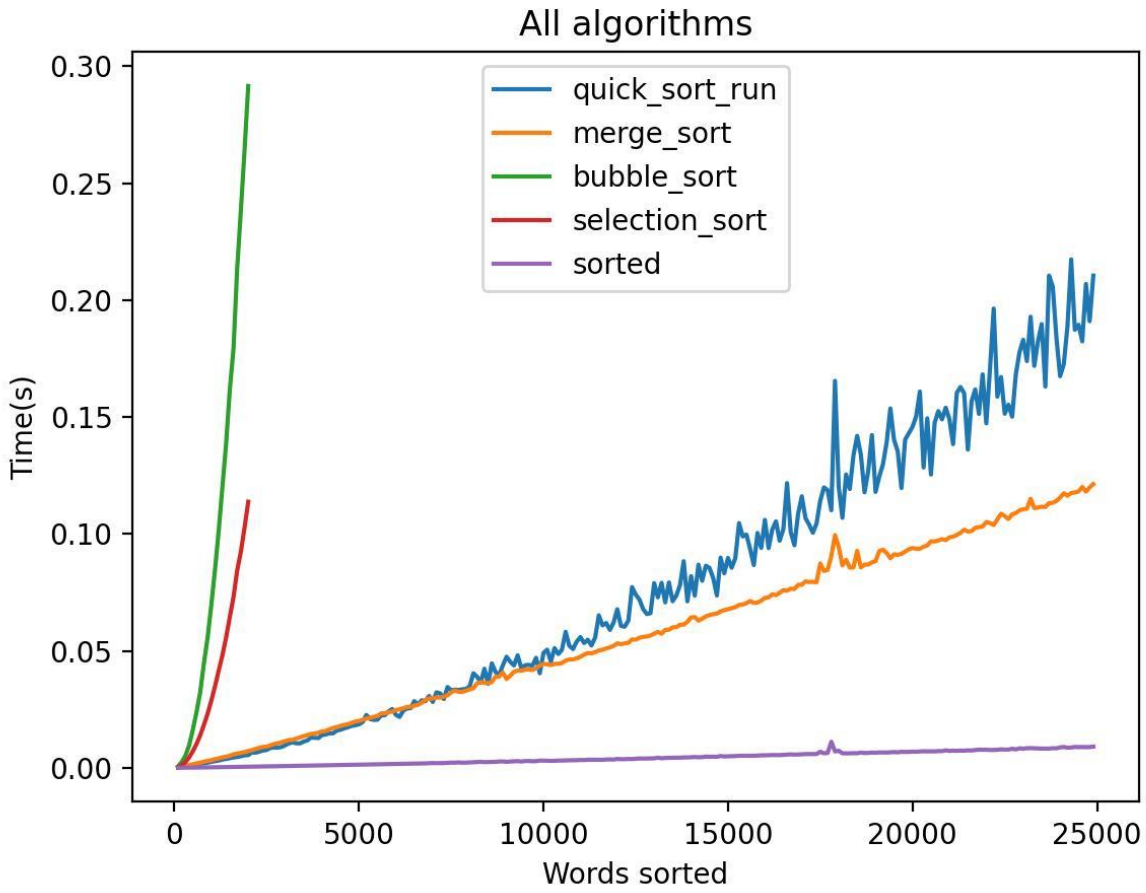
Quicksort



Ten algorytm również dzieli problem na mniejsze problemy. Tutaj lepiej widać jak przy dużych danych zaczyna odbiegać od funkcji liniowej. Jest trochę wolniejszy niż merge sort, lecz możlwe jest przeanalizowanie danych pod kątem optymalizacji algorytmu (lepsze wybieranie środkowego elementu podczas dzielenia). Z czasem zaobserwować można duże oscylacje, głównym powodem jest właśnie losowe wybieranie elementu środkowego.

Złożoność czasowa: nlog(n)

Wszystkie wykresy



Algorytmy O(n^2) zostały ograniczone do 2000 słów aby uchronić procesor przed spaleniem. Dodatkowo dodane jest porównanie z wbudowaną funkcją `sorted()`, 25 tysięcy danych nie robi na niej wrażenia. Wynika to z faktu, że jest zaimplementowana natywnie, więc nie ma spowolnienia pochodzącego od interpretera.

Kod źródłowy:

https://gitlab-stud.elka.pw.edu.pl/xxxxxx/22l-aisd-2-sort