AISDI – laboratorium z sortowania

Jakub Stępniak
Przemysław Wyziński
Lab 102, grupa 4

Cel ćwiczenia

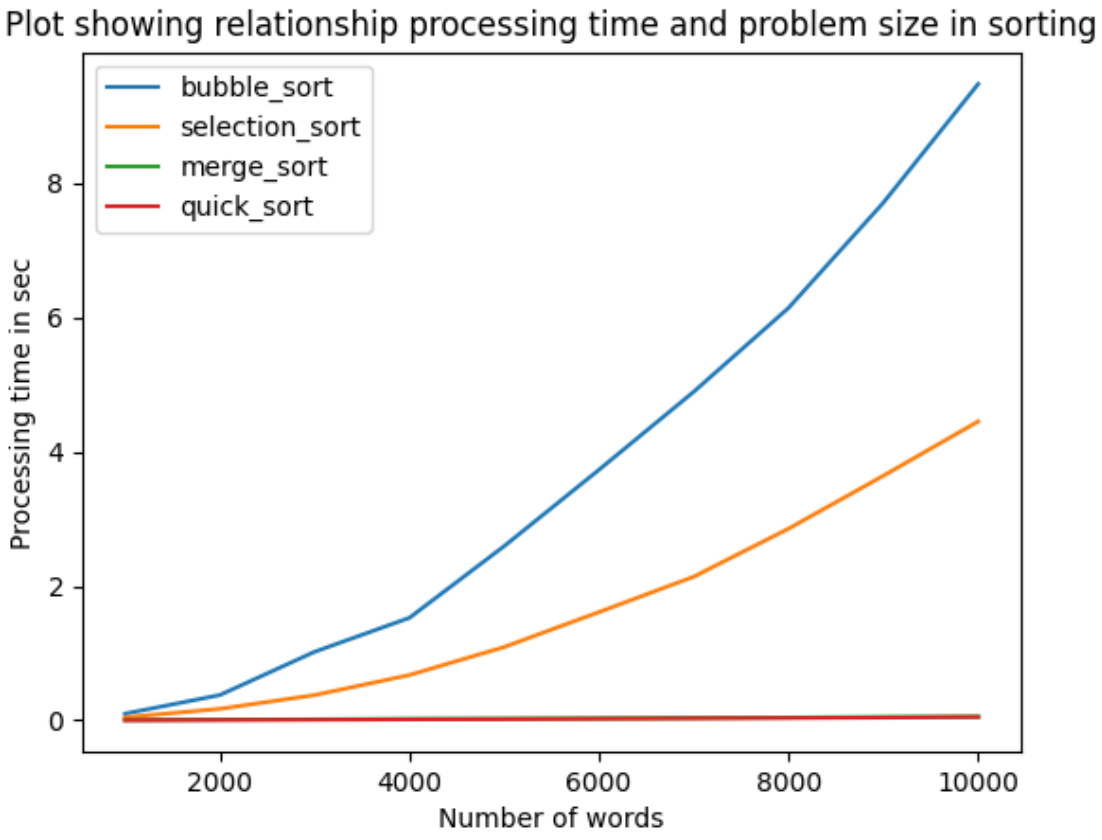
Celem ćwiczenia była implementacja i testowanie 4 algorytmów sortowania:

- Bubble sort
- Insertion sort
- Merge sort
- Quick sort

Testy algorytmów

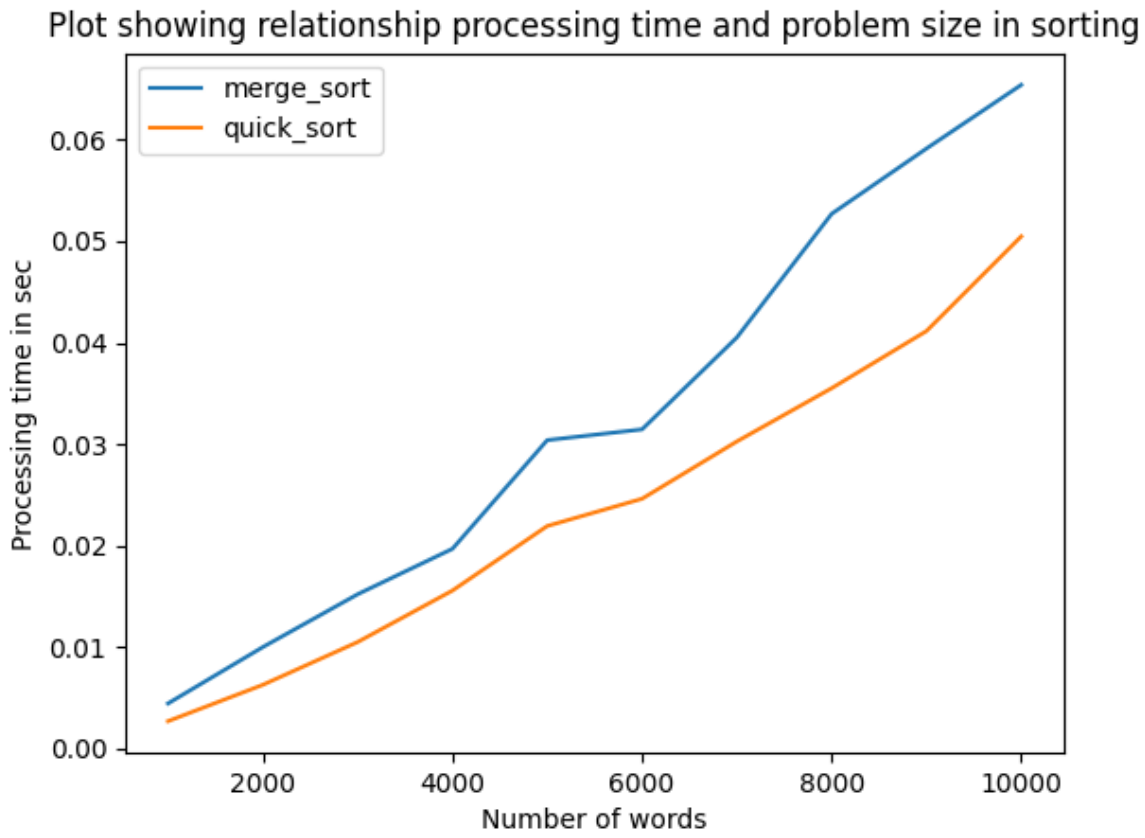
Testy polegały na zmierzeniu czasu potrzebnego na przesortowaniu n pierwszych słów wczytanych z pliku pan-tadeusz-unix.test dla n = 1000, 2000, …, 10000.

Otrzymany wykres:



W takiej skali przebieg algorytmów Merge sort oraz Quick sort jest nierozróżnialny, dlatego doświadczenie powtórzyliśmy z pominięciem wolnych algorytmów.

Wykres tylko dla algorytmów Merge sort oraz Quick sort:



Porównanie wyników z teoretycznymi zależnościami czasowymi

Dla algorytmów Bubble sort oraz Insertion sort teoretyczna złożoność czasowa to $O(n^2)$.

Przykładowe zmierzone czasy tych algorytmów zamieszczone są w tabeli:

n pierwszych słów	Bubble sort	Insertion sort
1000	99 ms	45 ms
2000	377 ms	172 ms
4000	1528 ms	675 ms

W przypadku Bubble sort zwiększenie problemu 2-krotnie wydłużyło czas działania algorytmu 3,81 razy oraz 4,05 razy.

W przypadku Insertion sort zwiększenie problemu 2-krotnie wydłużyło czas działania algorytmu 3,82 razy oraz 3,92 razy.

Dla algorytmów o złożoności czasowej $O(n^2)$ 2-krotne zwiększenie problemu powinno wydłużać czas działania algorytmu 4-krotnie, co w przybliżeniu zgadza się z doświadczeniami.

Złożoności czasowe mówią tylko o czynniku dominującym stąd w praktyce Insertion sort okazał się około 2 razy szybszy niż Bubble sort.

Dla algorytmów Merge sort oraz Quick sort teoretyczna złożoność czasowa to $O(n \log n)$.

Przykładowe zmierzone czasy tych algorytmów zamieszczone są w tabeli (dane z eksperymentu pierwszego):

n pierwszych słów	Merge sort	Quick sort
1000	4,33 ms	2,74 ms
2000	10,26 ms	6,11 ms
4000	21,91 ms	14,59 ms

W przypadku Merge sort zwiększenie problemu 2-krotnie wydłużyło czas działania algorytmu 2,37 razy oraz 2,14 razy.

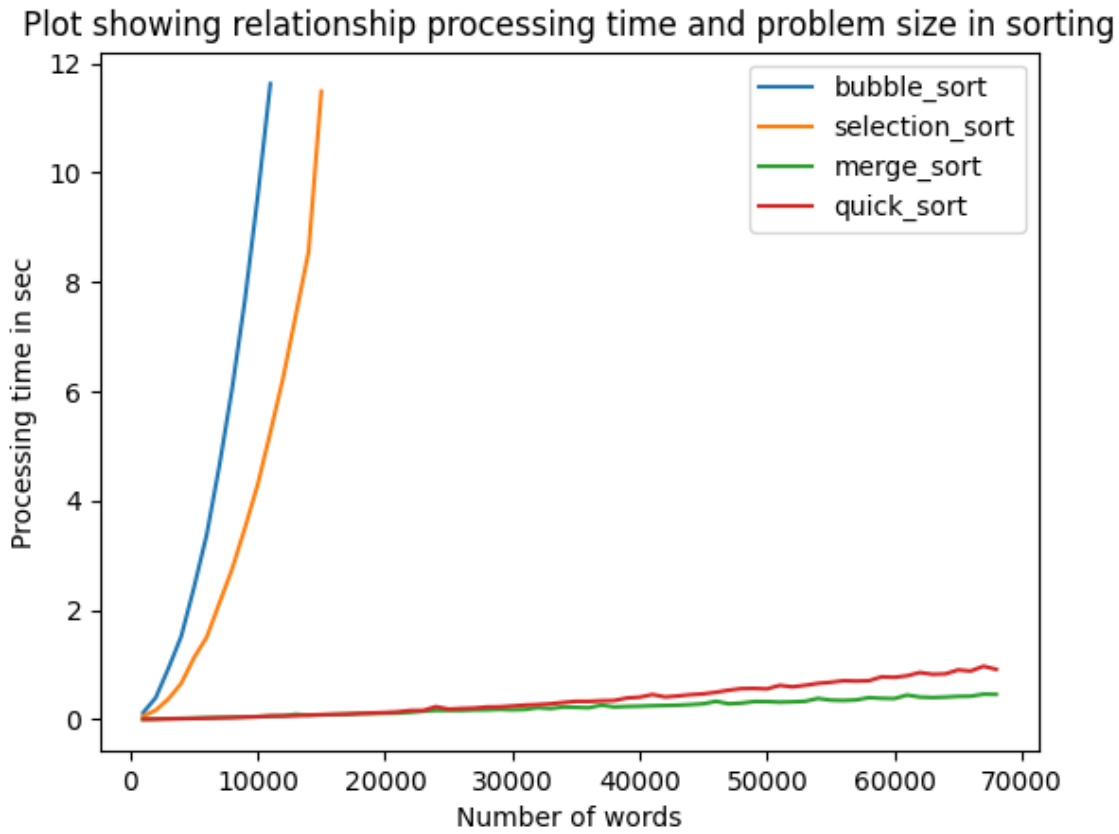
W przypadku Quick sort zwiększenie problemu 2-krotnie wydłużyło czas działania algorytmu 2,23 razy oraz 2,39 razy.

Dla algorytmów o złożoności czasowej $O(n \log n)$ 2-krotne zwiększenie problemu powinno wydłużać czas działania algorytmu niewiele bardziej niż ponad 2-krotnie, co również w tym przypadku zgadza się z doświadczeniami. W eksperymentach delikatnie szybszy okazał się Quick sort, jednak dla przypadku bardziej pesymistycznego mogłoby się okazać, że lepszy byłby Merge sort.

Test dodatkowy

Przesortowanie całego badanego pliku dla algorytmów o złożoności czasowej $O(n^2)$ okazało się bardzo trudnym zadaniem, podczas gdy algorytmy o złożoności czasowej $O(n \log n)$ poradziły sobie dobrze.

Wykres czasu trwania algorytmów w zależności od liczby słów:



Algorytmom Quick sort i Merge sort udało się przesortować wszystkie słowa poniżej 1s. W tym przypadku Merge sort okazał się szybszy od algorytmu Quick sort. Algorytmy Bubble sort oraz Selection sort zostały przerwane, gdyż wymagałyby zbyt dużej ilości czasu w celu przesortowania całego pliku.

Wnioski

Dla dużych problemów istnieje znacząca różnica pomiędzy złożonościami czasowymi $O(n^2)$ oraz $O(n \log n)$.

Czasy wykonania wewnątrz tych samych złożoności czasowych mogą się od siebie różnić, gdyż złożoność czasowa uwzględnia tylko czynnik dominujący, dlatego różne algorytmy o tej samej złożoności mogą wykonywać się w różnym czasie.