Algorytmy i Struktury Danych
Laboratorium nr 2- Algorytmy Sortowania

Autorzy:

Dominik Wiącek

Ivan Ziubanav

1.Przedstawienie i analiza algorytmów

a) Selection sort

W każdej iteracji zewnętrznej pętli iterujemy po kolejnych elementach tablicy, porównując je z aktualnie ekstremalną (najmniejszą lub największą) aż nie dojdziemy do końca tablicy. Po dojściu do końca posiadamy element o wartości ekstremalnej. Zapisujemy go, usuwamy z tablicy i znajdujemy kolejny ekstremalny. Tym samym wykonujemy n^2 operacji, gdzie n-ilość elementów do posortowania.

b) Merge sort

Algorytm działa rekurencyjnie. Dzieli on tablicę na podtablice (około) dwukrotnie mniejsze aż podtablice nie będą jednoelementowe (a więc posortowane). Następnie łączy on ze sobą kolejno podtablice aż do osiągnięcia ponownie pełnej tablicy, tym razem posortowanej. ###

c) Bubble sort

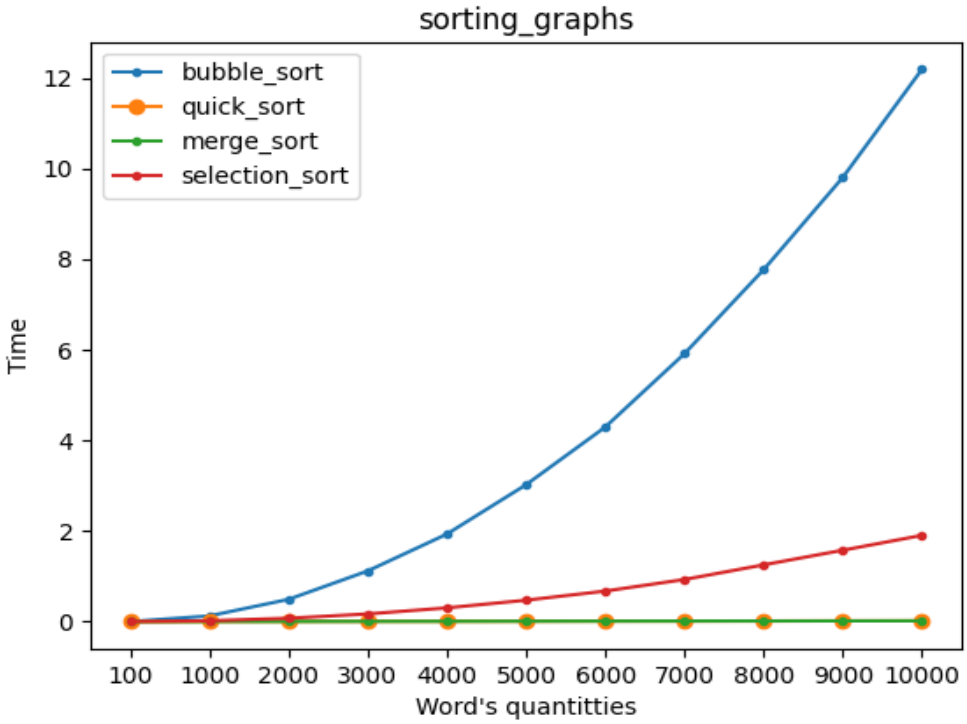
Algorytm zaczyna działać na początku tablicy. Porównuje on ze sobą pierwszy i następny element, jeżeli nie są w odpowiednim porządku (zależy czy sortujemy rosnąco czy malejąco) to zamieniamy je miejscami. Następnie przechodzi na drugi element i porównuje go z kolejnym itd. Po dojściu do końca tablicy wraca on na jej początek. Ten proces to jest jedna iteracja zewnętrznej pętli, gdzie wewnętrzną było przechodzenie po kolejnych elementach listy. Tym samym wykonujemy n*n (n^2) operacji (n=ilość elementów do posortowania). Wyjątkiem jest przypadek optymistyczny (kiedy elementy w tablicy są już posortowane), ponieważ wtedy iterujemy po zewnętrznej pętli tylko raz (aby dokonać wszystkich porównań co pozwoli nam określić, czy elementy są posortowane). Wtedy następuje wyłącznie n operacji.

d) Quick sort

Działa rekurencyjnie, bardzo podobnie do merge sortu. Polega on na ciągłym rozbijaniu tablicy na mniejsze podtablice w oparciu o wybrany element, zwany pivotem (w naszej implementacji jest to element środkowy tablicy/podtablicy). Do jednej podtablicy zostają dodane wszystkie elementy niemniejsze od wartości elementu pivot, a reszta do drugiej. Rozbijanie tablicy/podtablic odbywa się do momentu aż podtablica ma dokładnie 1 element (jest posortowana). Następnie następuje łączenie podtablic w coraz większe aż nie otrzymamy oryginalnej tablicy, ale posortowanej.

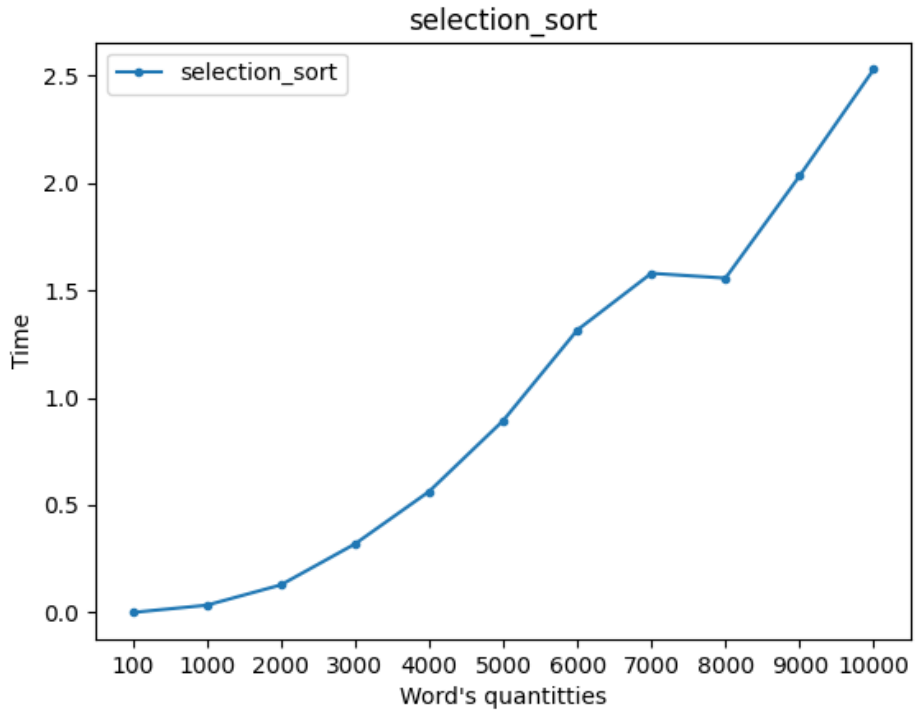
2. Wyniki pomiarów wydajności algorytmów:

a) Wykres porównawczy przedstawiający pomiary wydajności wszystkich analizowanych algorytmów

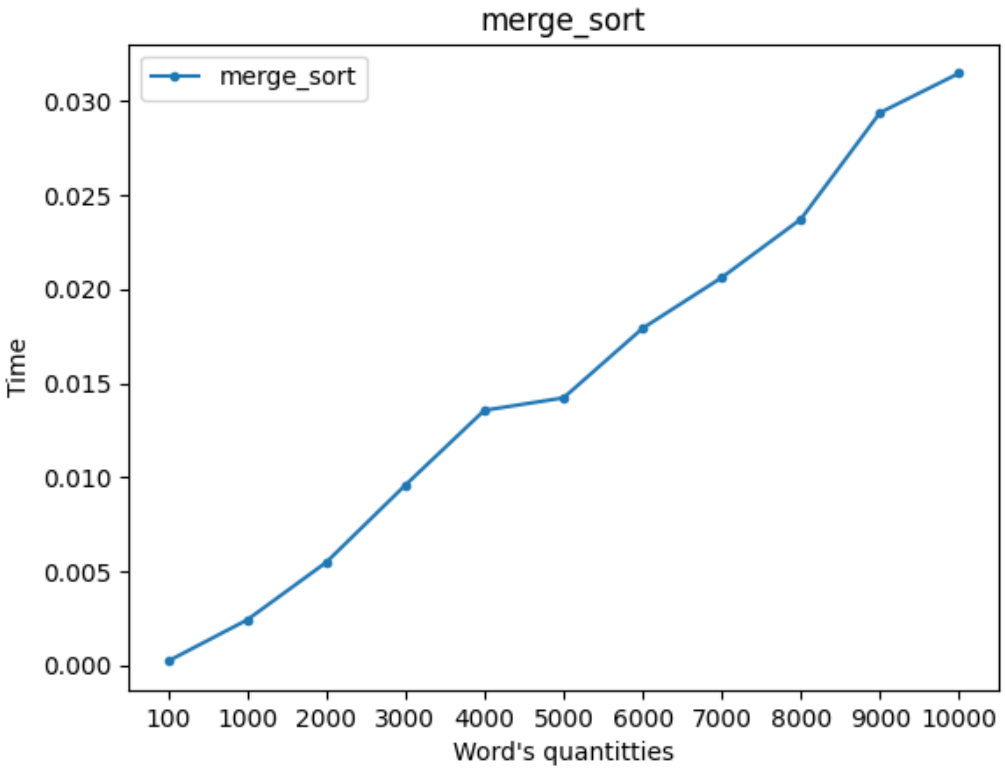


b) Wykresy przedstawiające pomiary wydajności poszczególnych algorytmów

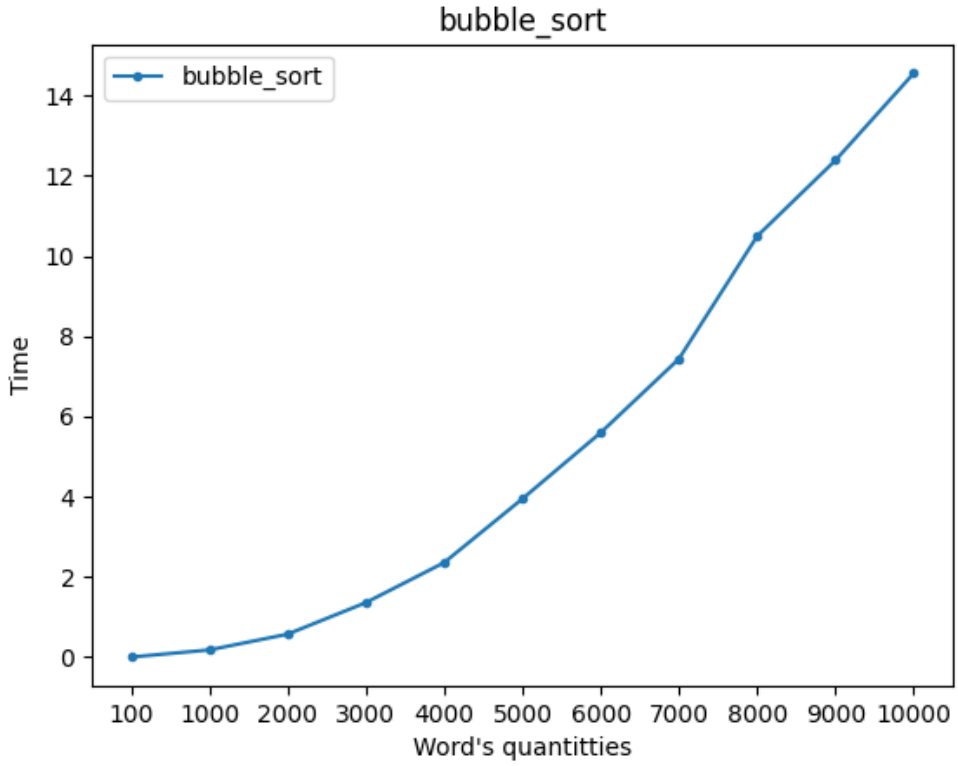
I. Selection sort



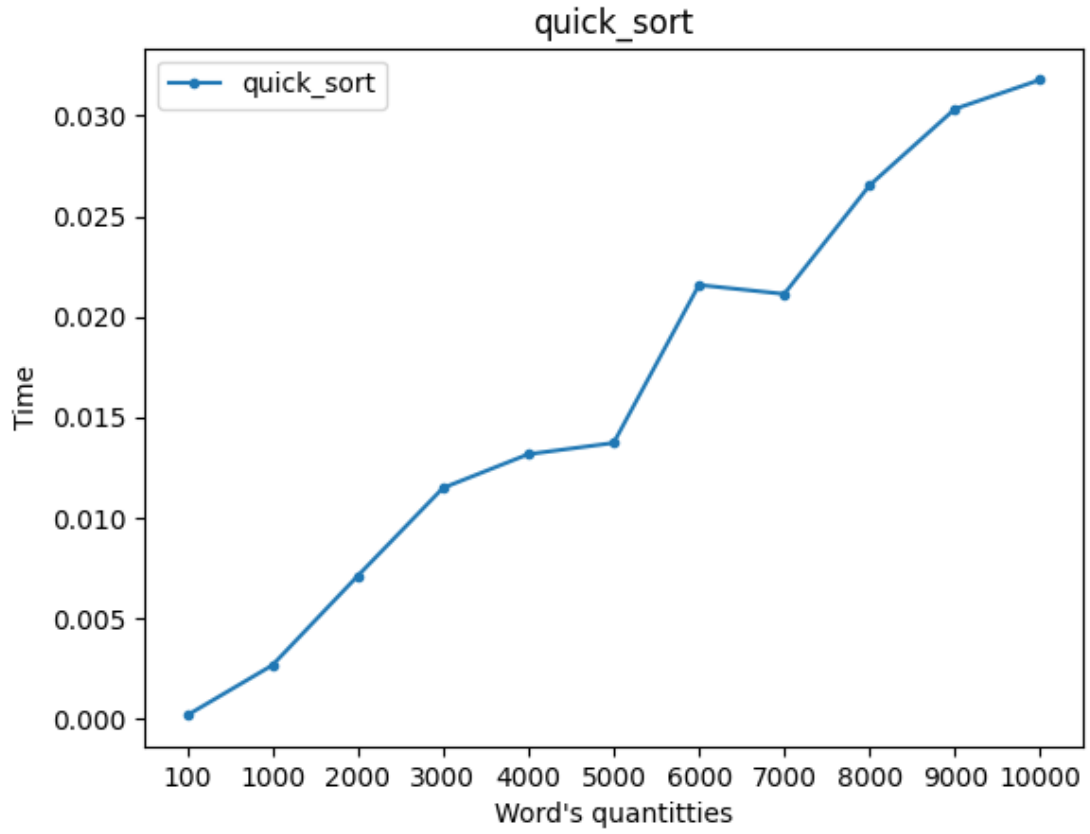
II. Merge sort



III. Bubble sort



IV. Quick sort



3. Wyniki: opis i interpretacja

Poza pewnymi anomaliami, których występowanie w przypadku danych losowych jest naturalne, można odczytać z wykresów, że wszystkie algorytmy zadziałały wedle przewidywań dla przypadków przeciętnych. Nie wystąpił ani przypadek optymistyczny dla bubble sort (co było do przewidzenia, w „Panu Tadeuszu" słowa nie są posortowane), ani pesymistyczny dla quick sort (z tego samego względu).

Z analizowanych algorytmów stabilne, a więc takie, które zwracają dane bez zamiany pozycji względnych elementów o tych samych wartościach, są selection sort, ponieważ w każdej iteracji znajduje najmniejszy element, jeżeli napotka element równy to pomija go. Stabilny jest również bubble sort (jeżeli porównuje 2 elementy o tych samych wartościach to nie zamienia ich). Merge sort także jest stabilny, ponieważ jeżeli w trakcie scalania dwóch tablic porównuje elementy z obu tablic o takiej samej wartości zawsze weźmie ten z tablicy lewej/prawej (zależy od wybranej implementacji) (u nas jest to tablica lewa). Jedynym algorytmem niestabilnym jest quick sort, ponieważ rozdziela on elementy względem wartości pivotu, a nie bierze pod uwagę oryginalnych pozycji elementów w tablicy.

Algorytmami in situ, a więc takimi, które wszystkie operacje realizują w ramach jednej tablicy, są bubble sort, ponieważ zamienia on wyłącznie miejscami sąsiadujące ze sobą elementy oraz selection sort, ponieważ znajduje on najmniejszy element (prócz tych już znalezionych) i przemieszcza je na odpowiednie miejsce w tej samej tablicy, w której jest reszta danych. Algorytmy merge sort oraz quick sort tworzą rekurencyjnie podtablice, przez co nie spełniają warunku bycia algorytmem in situ.