Algorytmy i struktury danych
Dariusz Walas
Grupa: zlinz_1_K37.2_Inf

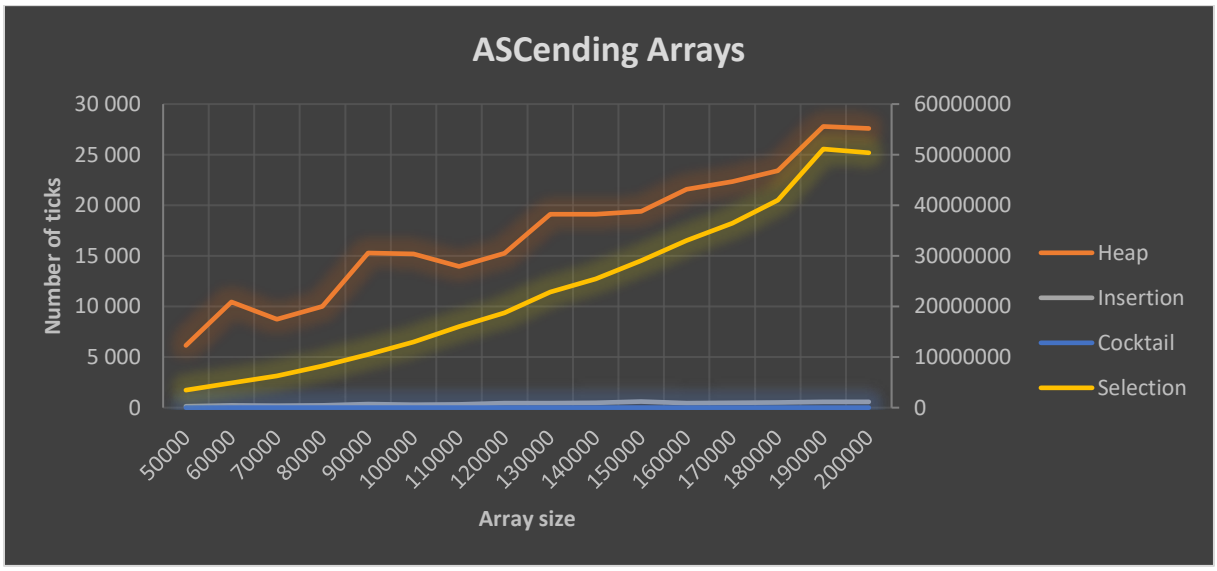
Projekt 3

Tematem pracy jest porównanie różnych algorytmów sortowania dla różnie zbudowanych tablic. Zadanie jest podzielone na trzy części. W pierwszej porównam różne metody sortowania dla danej budowy tablicy. W drugiej moja praca polegać będzie na sprawdzeniu z jaką tablicą lepiej sobie poradzi dana metoda sortowania. W ostatnim akapicie sprawdzimy czy algorytm Quicksort działa szybciej gdy jest napisany rekurencyjnie czy iteracyjnie dla tablicy z losowymi elementami. Na koniec sprawdzimy jakia metoda wyboru pivota w Quicksorcie lepiej się sprawdzi dla tablicy A-kształtnej.

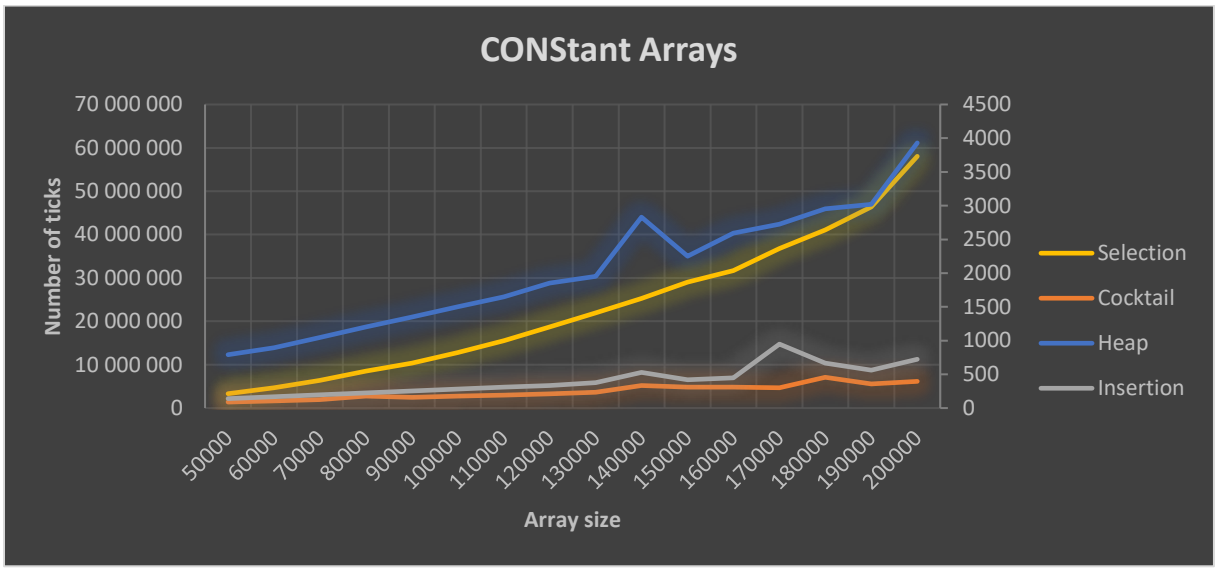
Część pierwsza

W tej części porównamy algorytmy sortowania dla danej budowy tablicy przekazanej do posortowania.

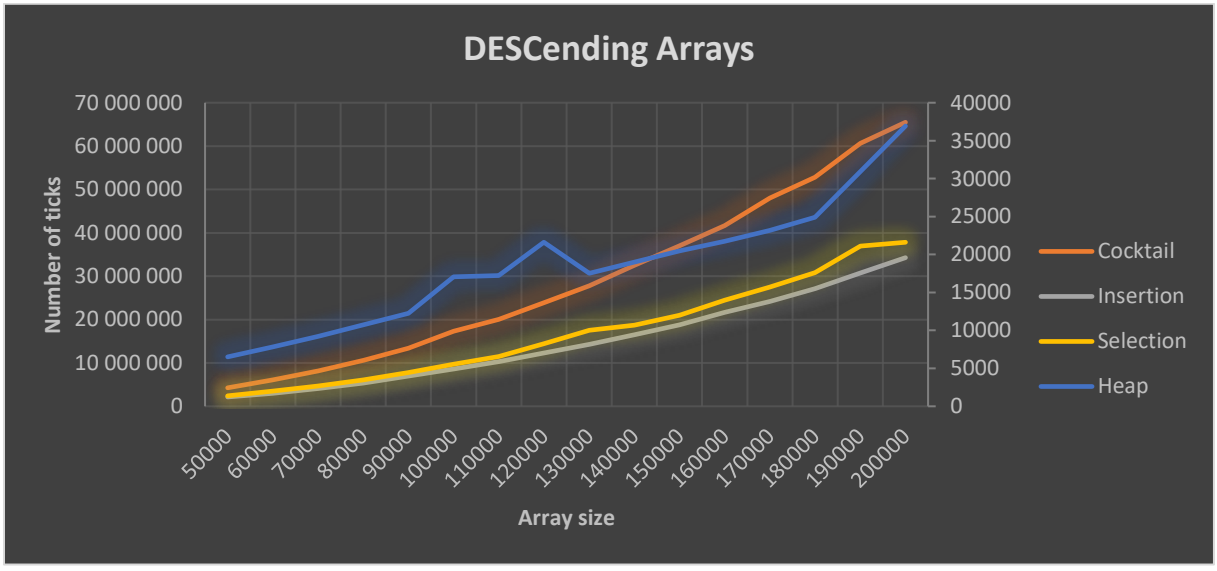
- Tablica posotowana rosnąco
    - Najszybszy algorytm: Cocktail sort
    - Najwolniejszy algorytm: Selection sort



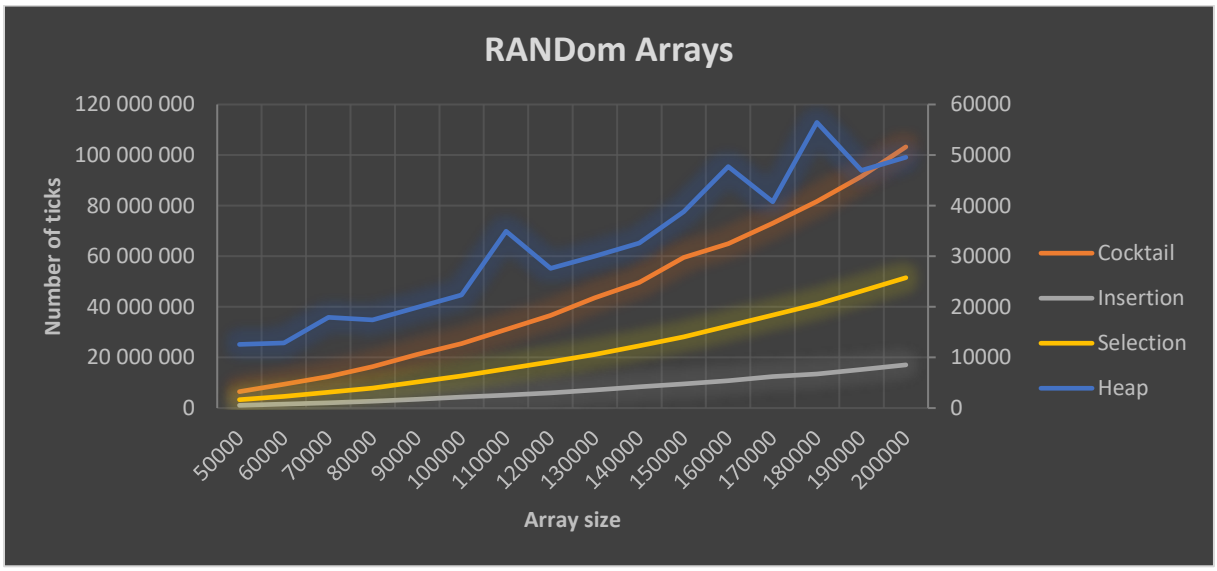
- Tablica stała
    - Najszybszy algorytm: Cocktail sort
    - Najwolniejszy algorytm: Selection sort



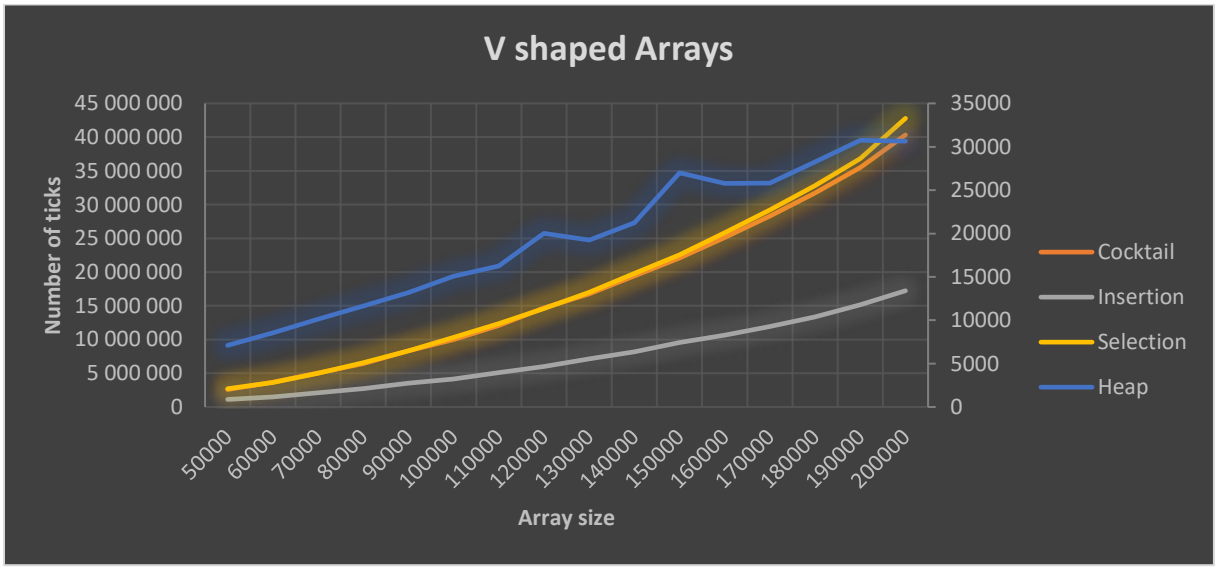
- Tablica posortowana malejąco
    - Najszybszy algorytm: Heap sort
    - Najwolniejszy algorytm: Cocktail sort



- Tablica z elementami losowymi
    - Najszybszy algorytm: Heap sort
    - Najwolniejszy algorytm: Cocktail sort



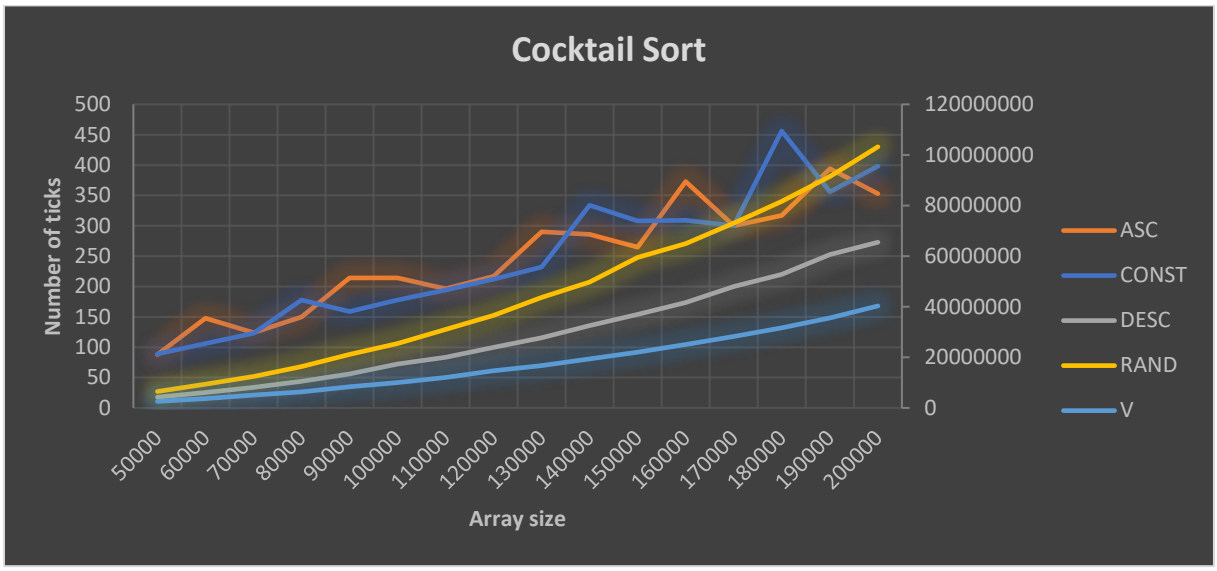
- Tablica V kształtna
    - Najszybszy algorytm: Heap sort
    - Najwolniejszy algorytm: Selection sort



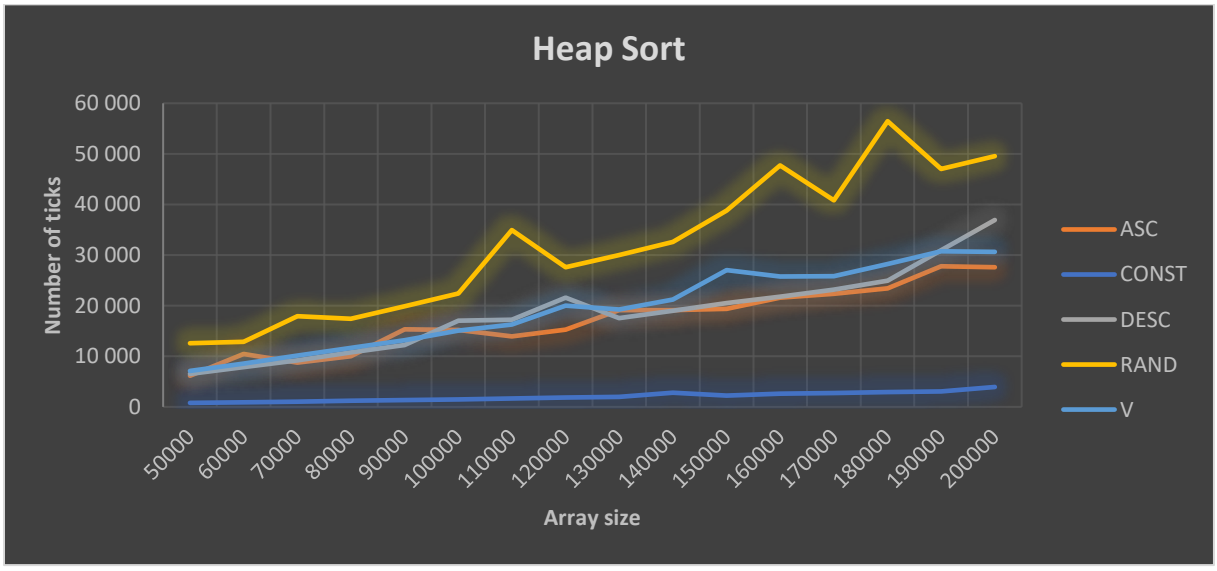
Część druga

W tej części porównamy z jakim rodzajem tablicy dany algorytm radzi sobie najlepiej, a z jakim najgorzej.

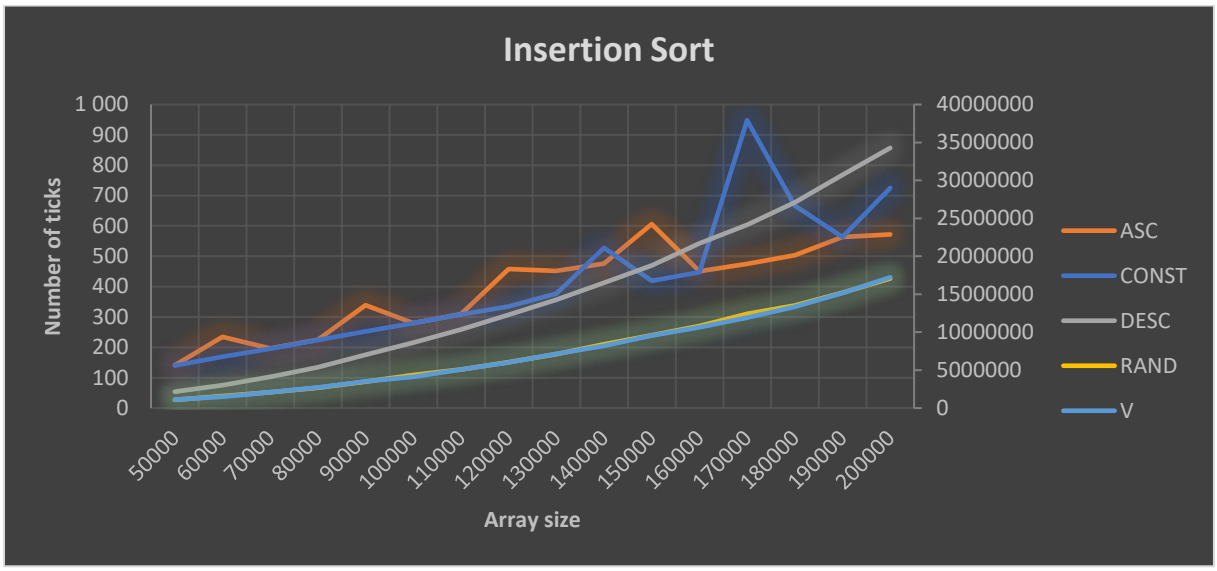
- Cocktail sort
  - Najszybciej posortowana tablica: rosnąca
  - Najwolniej posortowana tablica: losowa



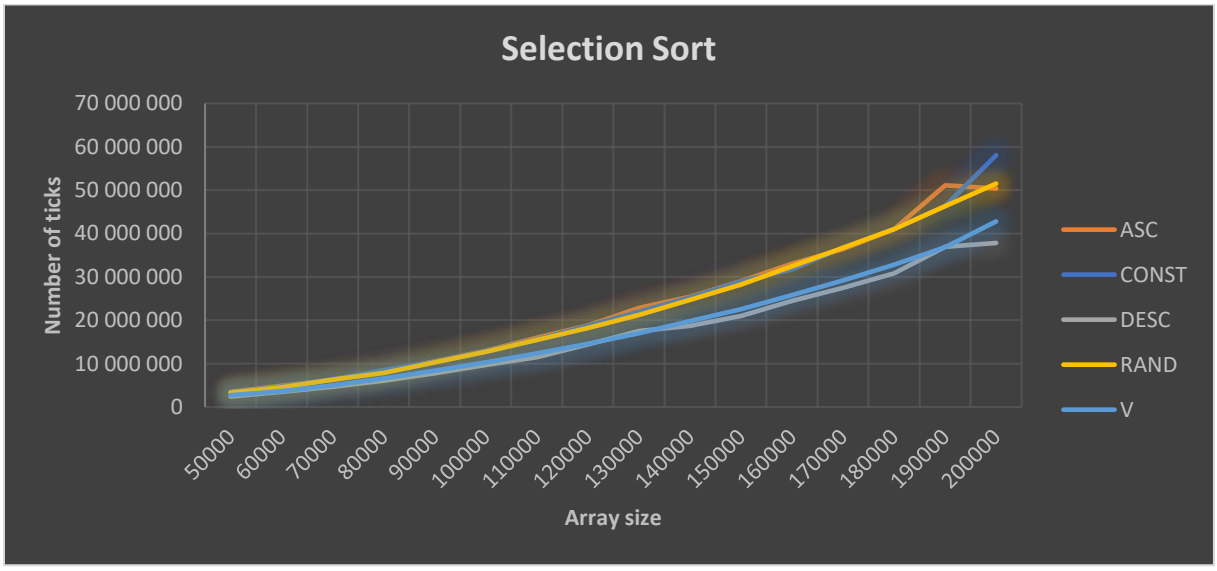
- Heap sort
  - Najszybciej posortowana tablica: stała
  - Najwolniej posortowana tablica: losowa



- Insertion sort
    - Najszybciej posortowana tablica: rosnąca
    - Najwolniej posortowana tablica: malejąca

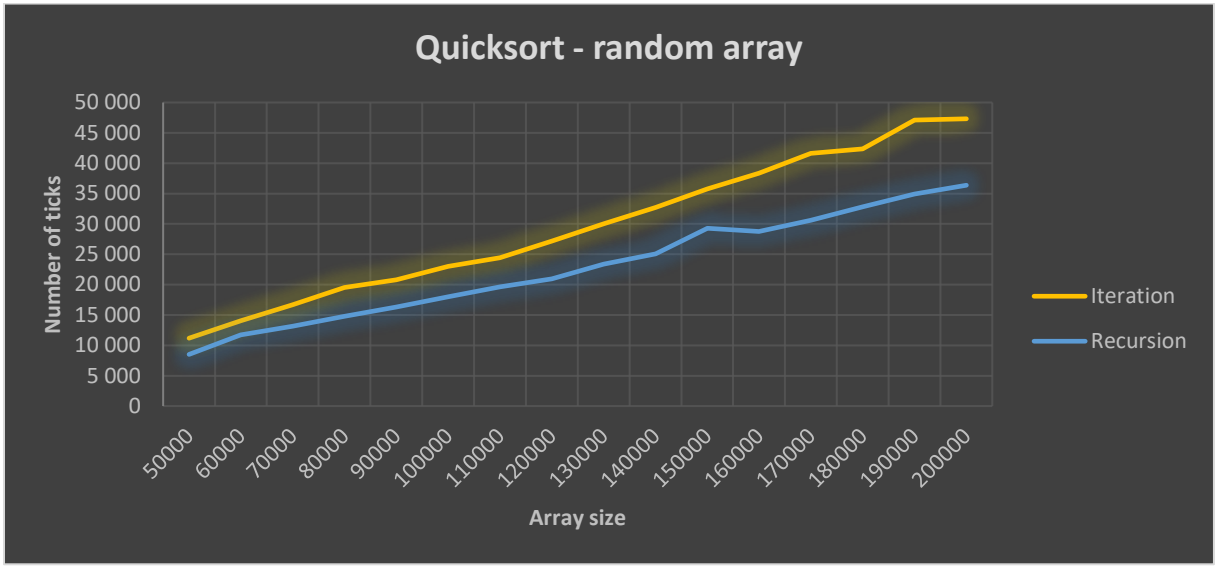


- Selection sort
    - Najszybciej posortowana tablica: malejąca
    - Najwolniej posortowana tablica: stała



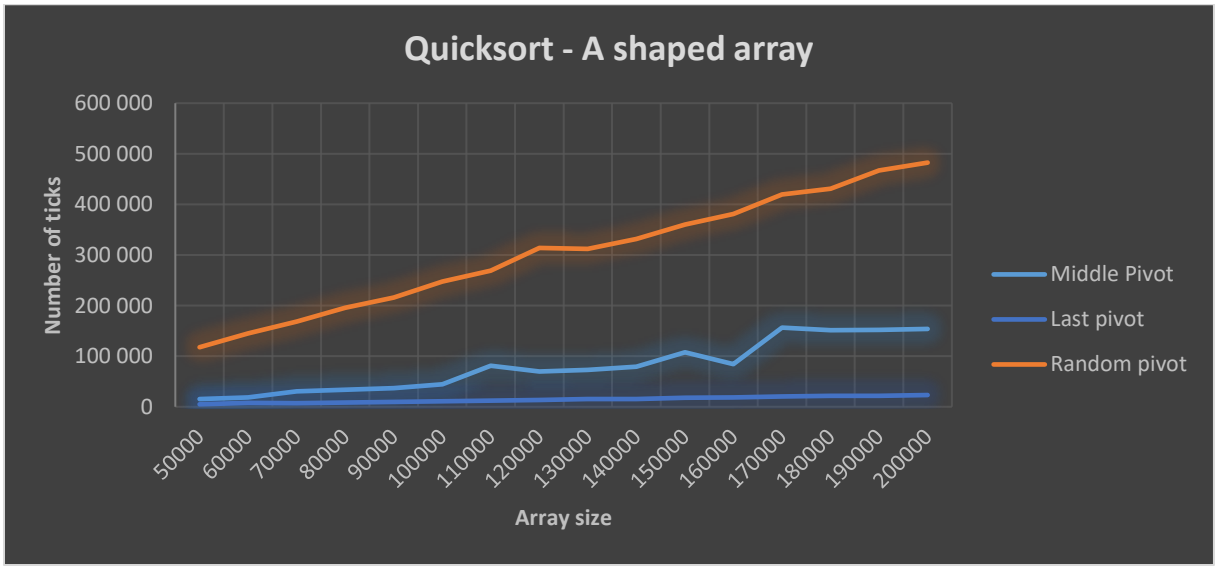
Część trzecia

W tej części należało porównać czy algorytm sortowania Quick sort zaimplementowany iteracyjnie jest szybszy od algorytmu rekurencyjnego. Do testu wykorzystano tablicę losowych elementów.



Jak pokazuje powyższy wykres algorytm napisany rekurencyjnie jest szybszy niż iteracyjny.

Ostatnim zadaniem było sprawdzenie szybkości sortowania tablicy A-kształtnej przez algorytm Quick sort (rekurencyjny). W tym ćwiczeniu wykorzystano 3 metody wyboru pivota (klucz podziału) – środkowy, ostatni i losowy.



Jako pokazuje wykres dla tego zestawnia najlepszy wybór pivota to ostatnie miejsce w tablicy.