Paweł Pawlaczyk nr. Albumu 102876

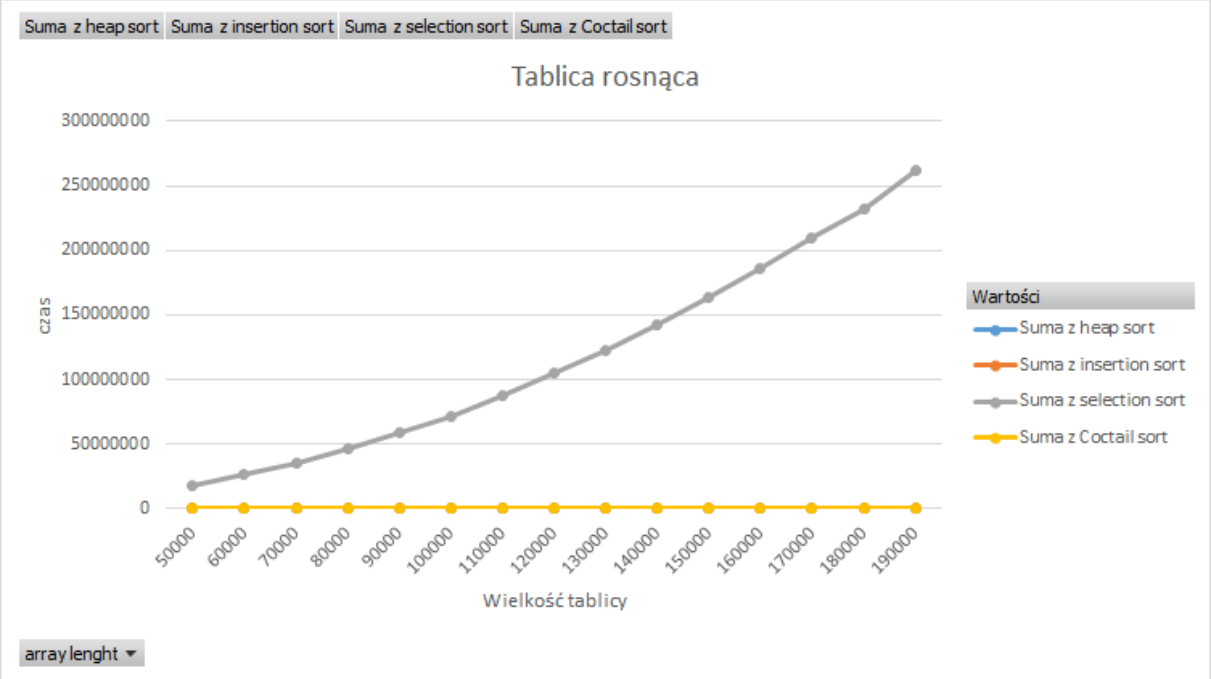
Projekt 3

W tym projekcie zajmę się badaniem różnych algorytmów sortowania pod względem tablic jakie te algorytmy musza posortować oraz ich implementacji. W pierwszej i drugiej części pod lupę będą wzięte algorytmy: Heap Sort, Insertion Sort, Selection Sort oraz Coctail Sort. Natomiast w części trzeciej będę rozpatrywał różne wariacje algorytmu Quick Sort.

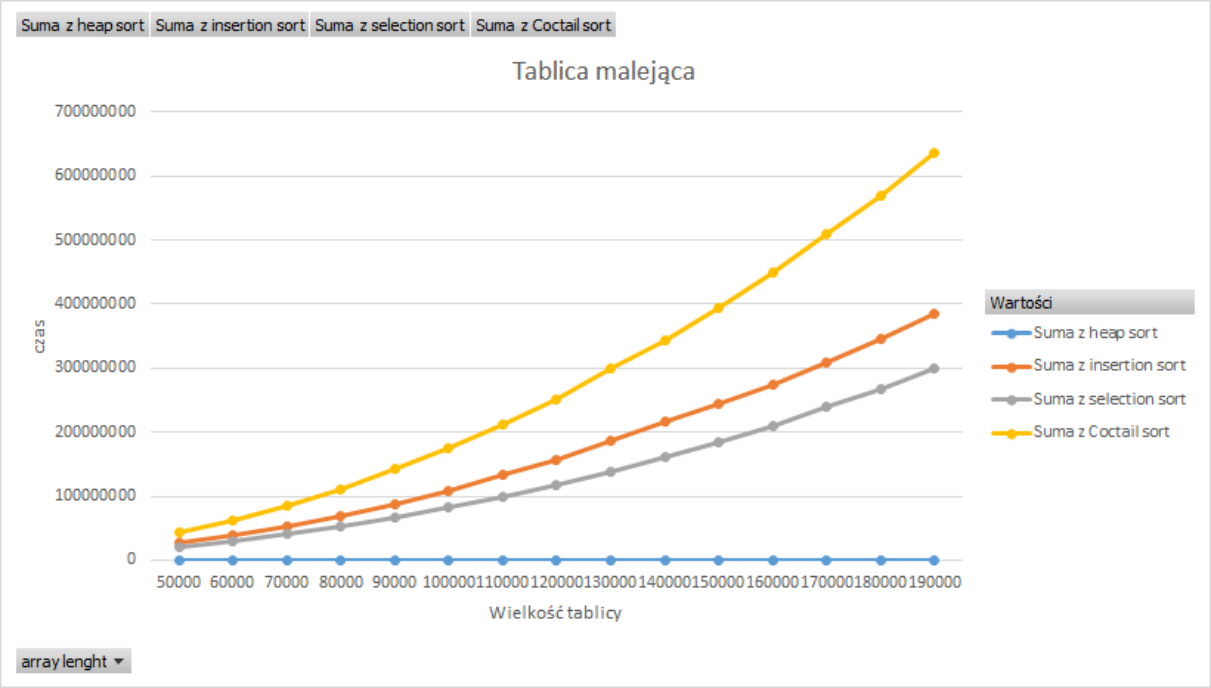
Pomiary zostały wykonane na jednostce operującej na 3GB pamięci podręcznej, oraz na procesorze: Intel Xeon X5450 3.00GHz.

Część I

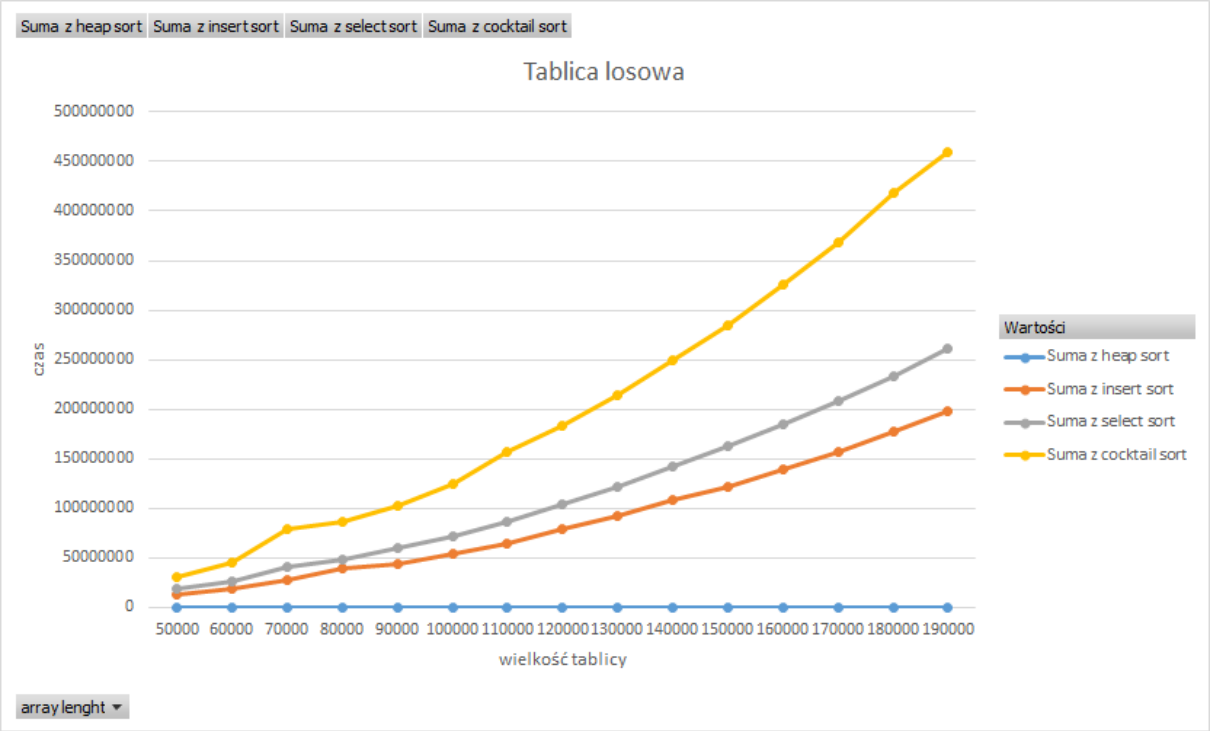
Pomiar ze względu na typ tablicy



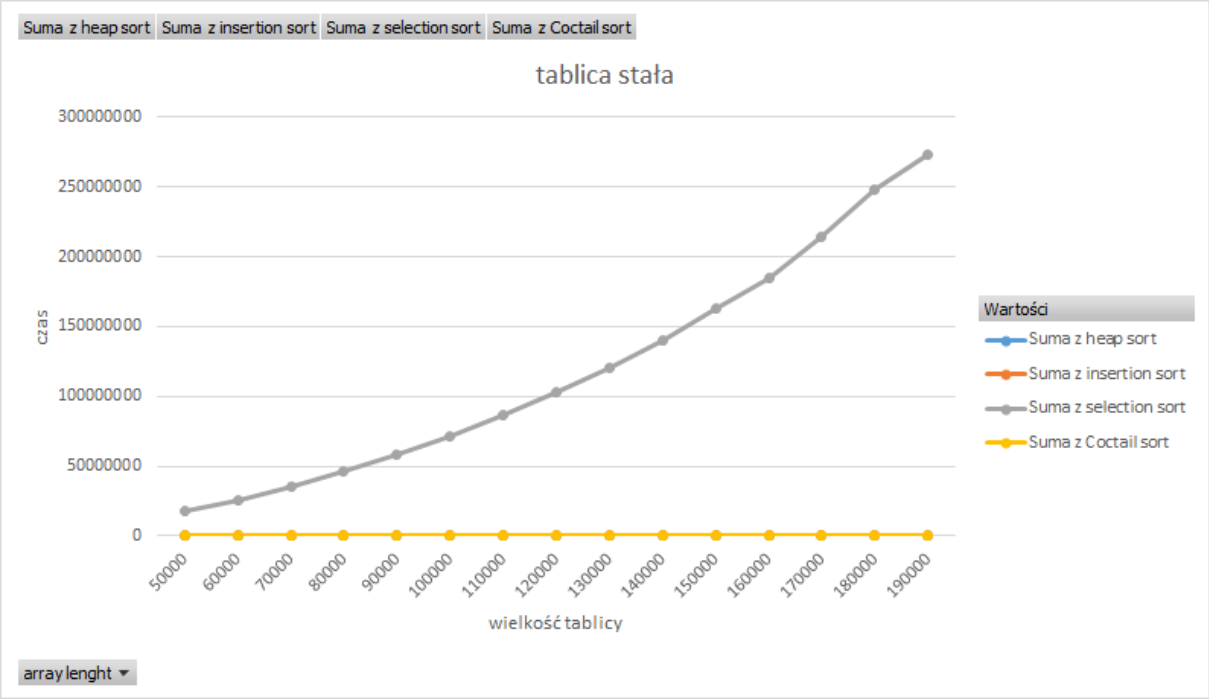
W przypadku tablicy rosnącej, trudno jest określić zwycięzcę na podstawie powyższego wykresu. Natomiast jest pewne, że nie należy użyć algorytmu Selection Sort, gdyż osiągnął najgorszy wynik, znacznie wyróżniając się na tle innych algorytmów.



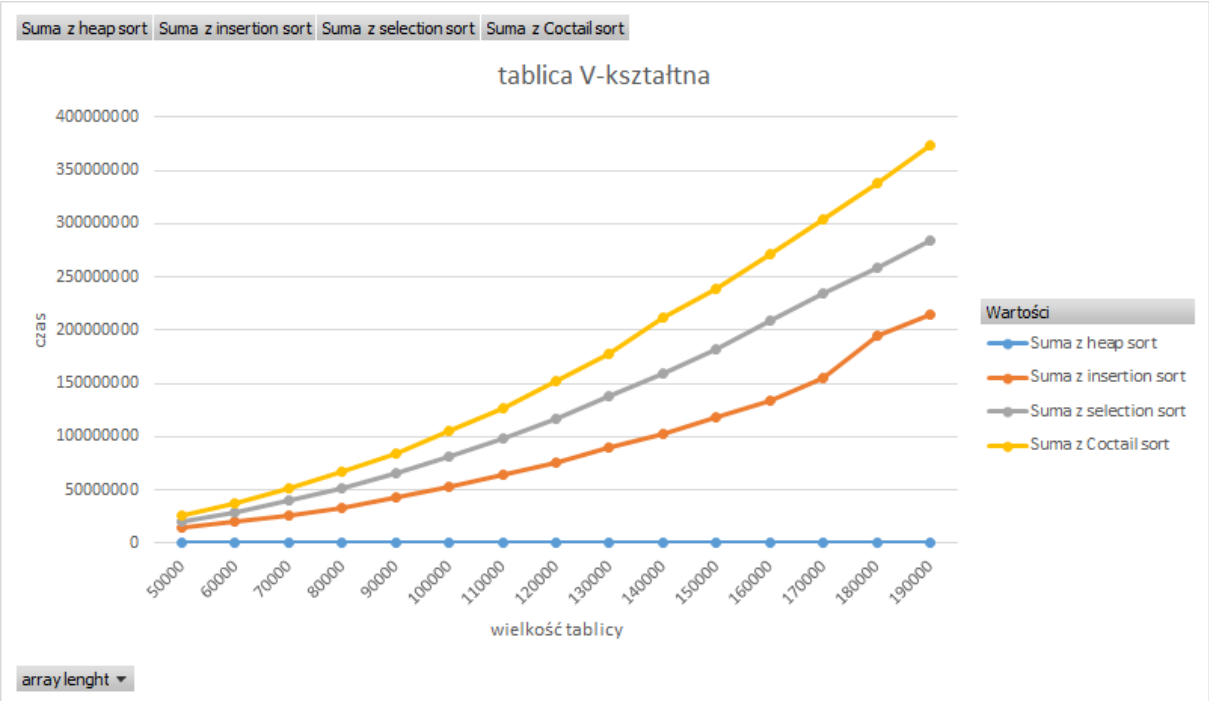
W przypadku tablicy malejącej, doskonale możemy zauważyć, że najlepiej sprawdzi algorytm Heap Sort. Natomiast Coctail Sort wypada bardzo mizernie w tym zestawieniu. Lepiej w takim zastosowaniu go nie używać.



W przypadku losowej tablicy wynik jest taki sam jak wyżej. Heap Sort sprawdzi się najlepiej natomiast Cocktail Sort, będzie kompletnie nie efektywny w tym zastosowaniu.



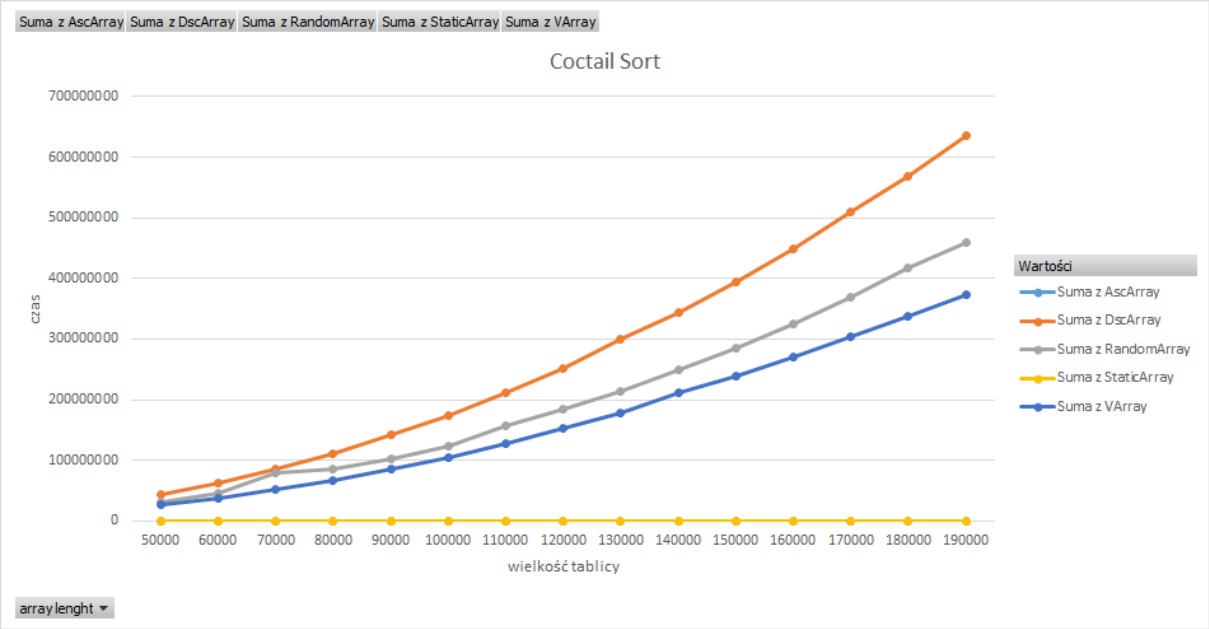
W przypadku tablicy stałej, trudno jest określić najlepszy algorytm. Natomiast łatwo jest wyłonić najgorszy. Jest to oczywiście Selection Sort.



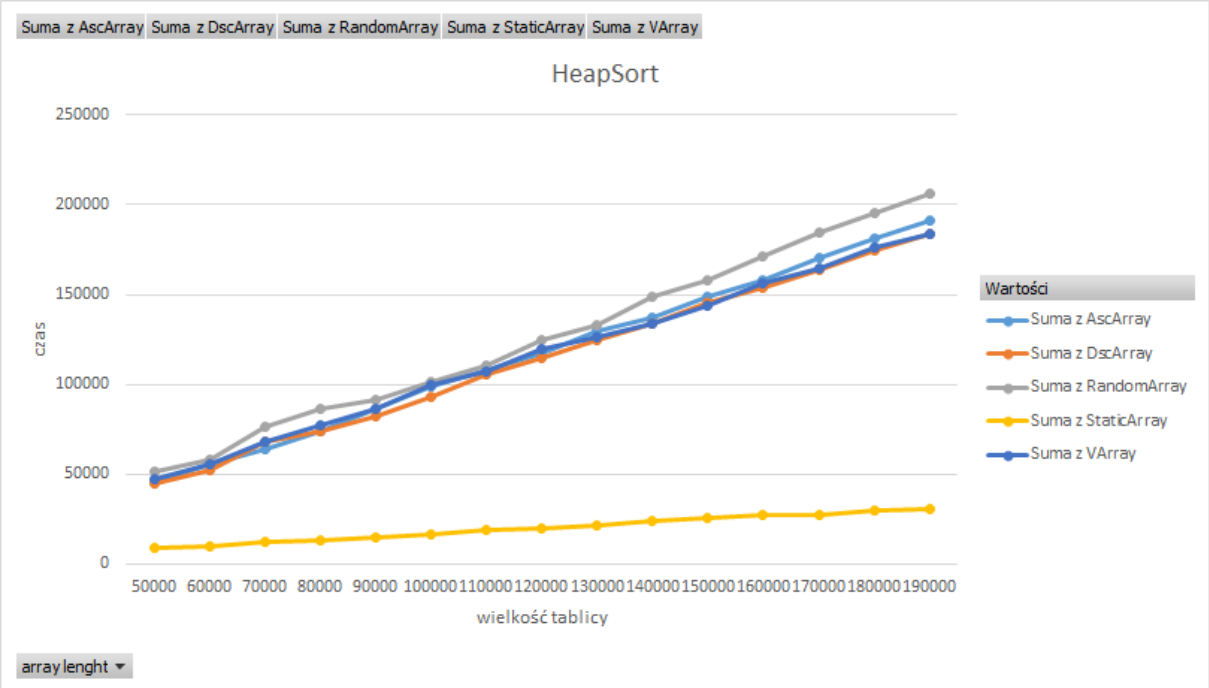
W przypadku tablicy V-kształtnej, możemy ponownie zauważyć ogromną dominację Heap Sort nad pozostałymi algorytmami. Najbardziej mizernie wypada ponownie Coctail Sort.

Część II

Pomiar ze względu na tym algorytmu



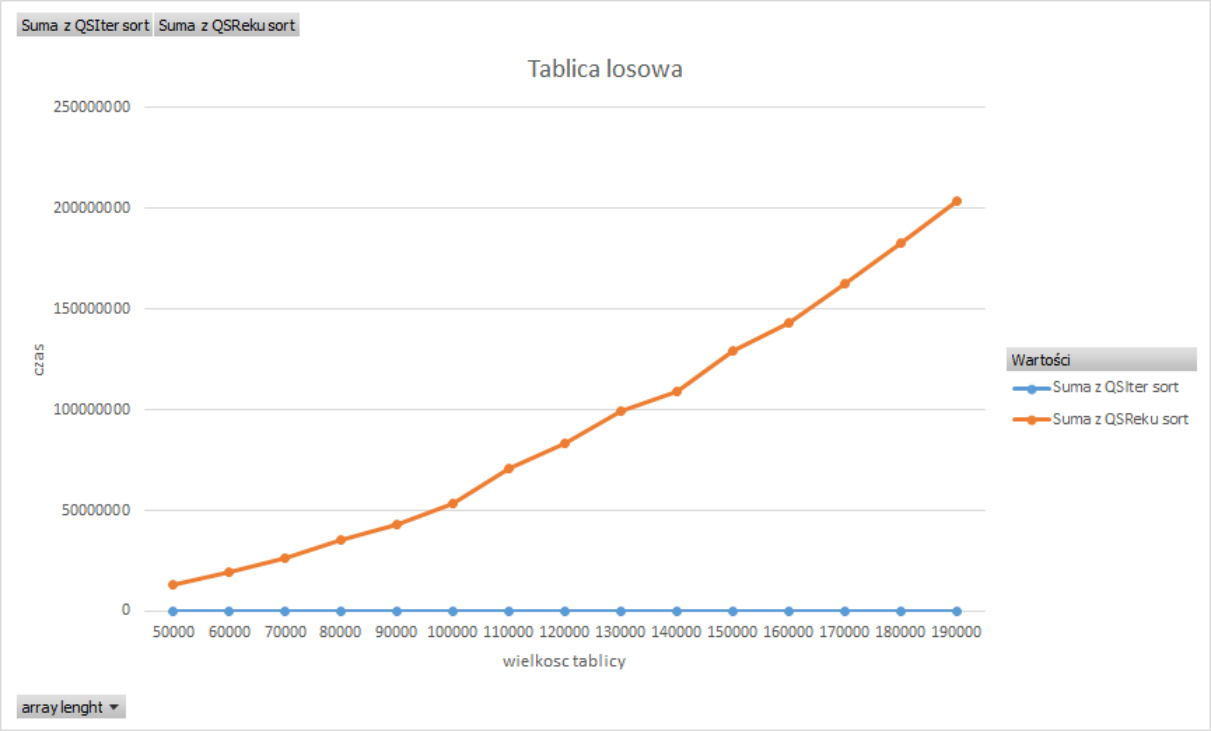
Coctail Sort najlepiej sobie radzi z tablicami stałymi, natomiast najbardziej negatywny przypadek jest wtedy, gdy natrafi na tablice posortowaną malejąco.



Heap Sort najlepiej się sprawdza w przypadku tablicy stałej. W pozostałych tablicach jego efektywność jest podobna, jednakże jak możemy zauważyć na powyższym wykresie, najgorszy przypadek przytrafia się wtedy, gdy trafi na tablice wypełnioną losowymi danymi.

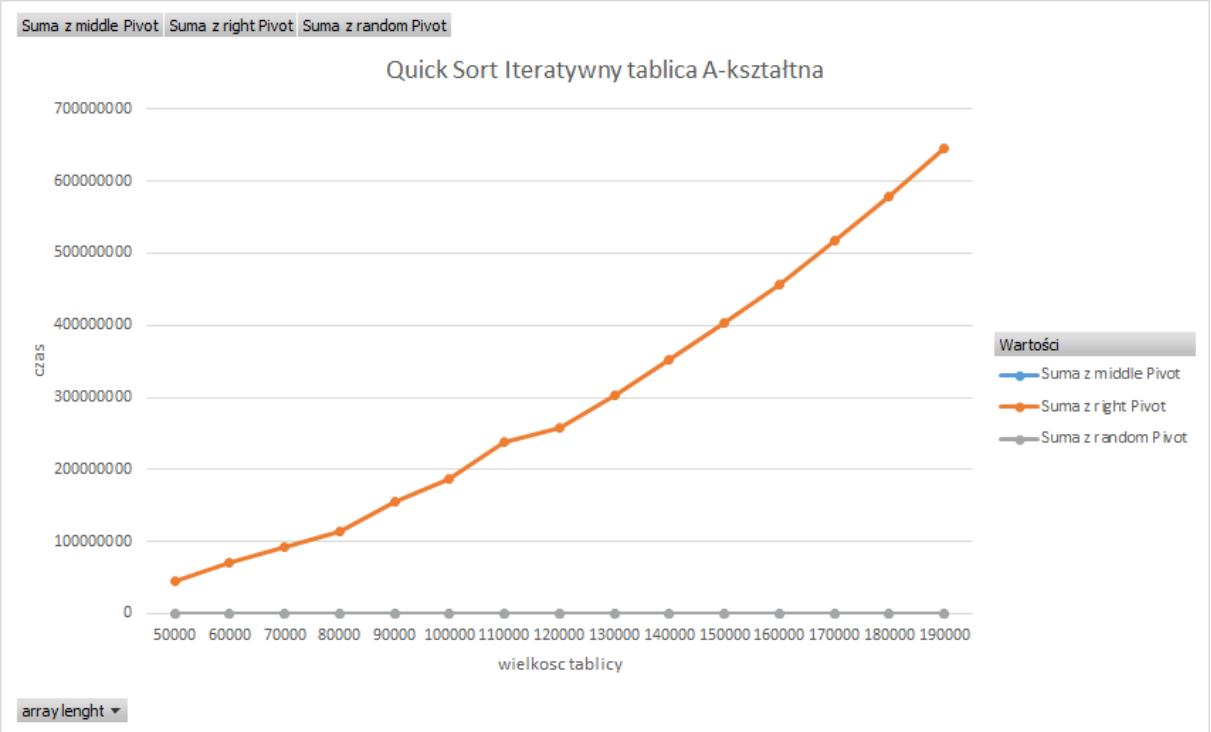
Część III

Quick Sort Rekurencyjny a Quick Sort Iteratywny



Powyższy wykres ukazuje dwa sposoby implementacji algorytmu Quick Sort. Jeden jest rekurencyjny a drugi iteratywny. Pomiar został wykonany na tablicy wypełnionej losowymi elementami. Doskonale widać, że iteracyjna implementacja algorytmu Quick Sort jest o wiele wydajniejsza niż implementacja rekurencyjna.

Quick Sort Iteratywny – różne pozycje pivota



Wykres ukazuje różnice w wydajności algorytmu Quick Sort w implementacji iteratywnej w zależności od położenia pivota, pracując na tablicy A-kształtej. Najgorszy jest przypadek, gdy ustawimy pivot na skraju sortowanej tablicy (w przypadku tego testu był to prawy skraj). Natomiast, gdy pivot będzie wybierany losowo, lub będzie stawiany na środku uzyskami o wiele lepsze rezultaty.

Podsumowanie

Projekt ten doskonale ukazał, że nie tylko rodzaj algorytmu sortującego jest ważny, ale również tablica, jaką ów algorytm musi posortować. Te różnice doskonale widać w części drugiej projektu. Natomiast część trzecia doskonale nam uzmysławia, że sama logika działania algorytmu to nie wszystko. Kluczowy jest również sposób implementacji algorytmu, oraz sposób wyznaczania punktu odniesienia (pivotu).