Algorytmy sortowania

Nr ćwiczenia: 2

Temat ćwiczenia: Sortowania

Nazwisko i Imię prowadzącego kurs:

Wykonawca:	Mgr inż. Marcin Ochman
Imię i Nazwisko nr indeksu	Martyna Żukowska 252877
Termin zajęć: dzień tygodnia, godzina	Czwartek 15:15
Numer grupy ćwiczeniowej	E12-99j
Data oddania sprawozdania:	22.04.2021
Ocena końcowa	

1. Cel ćwiczeń

Należy zaimplementować 3 z 6 wymienionych algorytmów sortowania (dwa o złożoności nie większej niż O(n log n)). Następnie należy zbadać ich wydajność dla różnych rozmiarów problemów. Sprawozdanie ma zawierać wykresy czasu wykonaniu algorytmu w zależności od liczby danych do posortowania.

2. Teoria

Sortowanie Szybkie jest jednym z algorytmów sortowania działających na zasadzie „dziel i zwyciężaj". Sortowanie to wybiera element rozdzielający na dwie podtablice i porównuje elementy z tym elementem rozdzielającym: dla elementów po lewej stronie dzielącego szuka się elementów większych, natomiast po prawej szuka się elementów mniejszych, następnie zamienia elementy ze sobą. Kończymy proces, gdy kolejny fragment uzyskany z podziału zawiera pojedynczy element. Złożoność obliczeniowa algorytmu Quicksort wynosi Ω(NlogN).

Sortowanie przez wstawianie to jeden z najprostszych algorytmów sortowania. Dzieli on nasz wektor na część uporządkowaną i nieuporządkowaną. Bierzemy pierwszy element z części nieuporządkowanej i wstawiamy do części uporządkowanej na odpowiednie miejsca. Złożoność obliczeniowa algorytmu insertSort wynosi Ω(N^2).

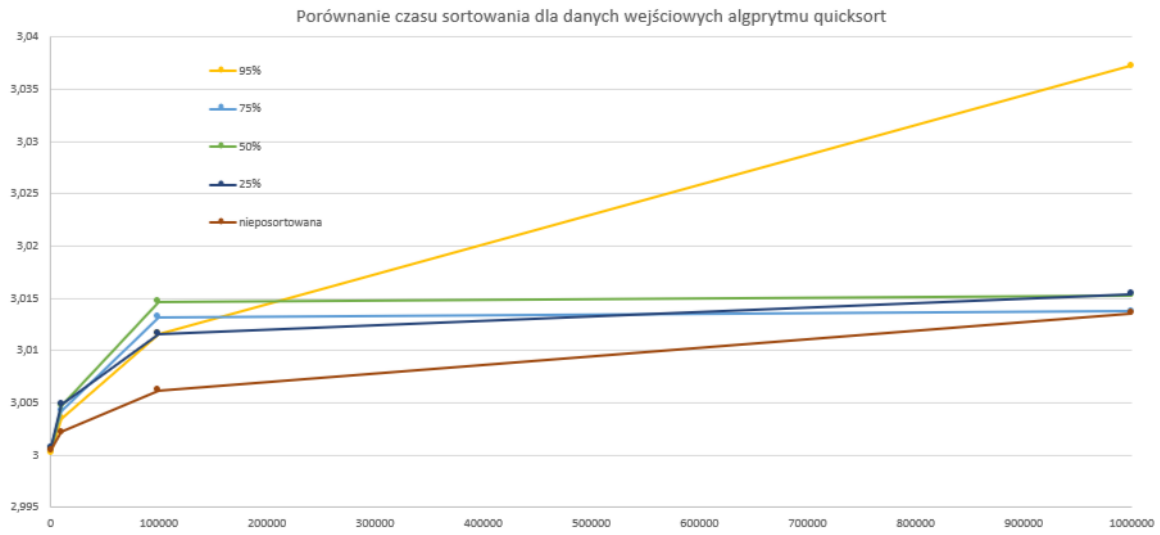
Sortowanie przez Scalanie jest jednym z algorytmów sortowania działających na zasadzie „dziel i zwyciężaj". Jest to sortowanie, gdzie dzieli się wektor dokładnie na połowę, i rekurencyjnie dzieli się podwektor aż do momentu otrzymania jednego elementu wektora. Następnie sortuje się każdy podwektor oddzielnie, później łączy się je i sortuje ponownie aż do otrzymania pełnego posortowanego wektora. Złożoność obliczeniowa algorytmu sortowania przez scalanie wynosi Ω(NlogN).

3. Eksperyment

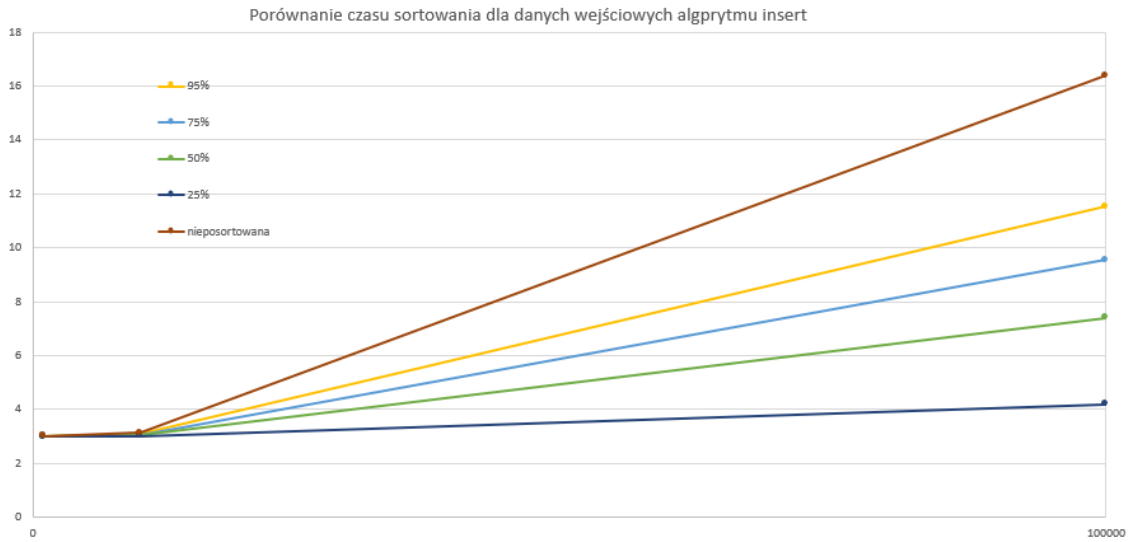
3.1 Tabela pomiarów

	95%	75%	50%	25%	całość
quicksort					
1000	3,00021405	3,0007414	3,0005214	3,0006914	3,00047
10000	3,0034914	3,0042314	3,0046814	3,0048514	3,00222
100000	3,0116114	3,0132114	3,0146914	3,0115914	3,00621
1000000	3,03721399	3,0137614	3,0152814	3,0153814	3,01359
scalanie					
1000	3,0059614	3,0060514	3,0059414	3,0060414	3,00615
10000	3,1527214	3,152414	3,1519314	3,1523614	3,15227
100000	18,570114	17,0414006	16,6491401	16,562614	17,6643
insert					
1000	3,0042914	3,0037814	3,0026214	3,0008814	3,00597
10000	3,0871814	3,0684614	3,0495814	3,0165914	3,13353
100000	11,533314	9,5456714	7,4063414	4,19641401	16,3784

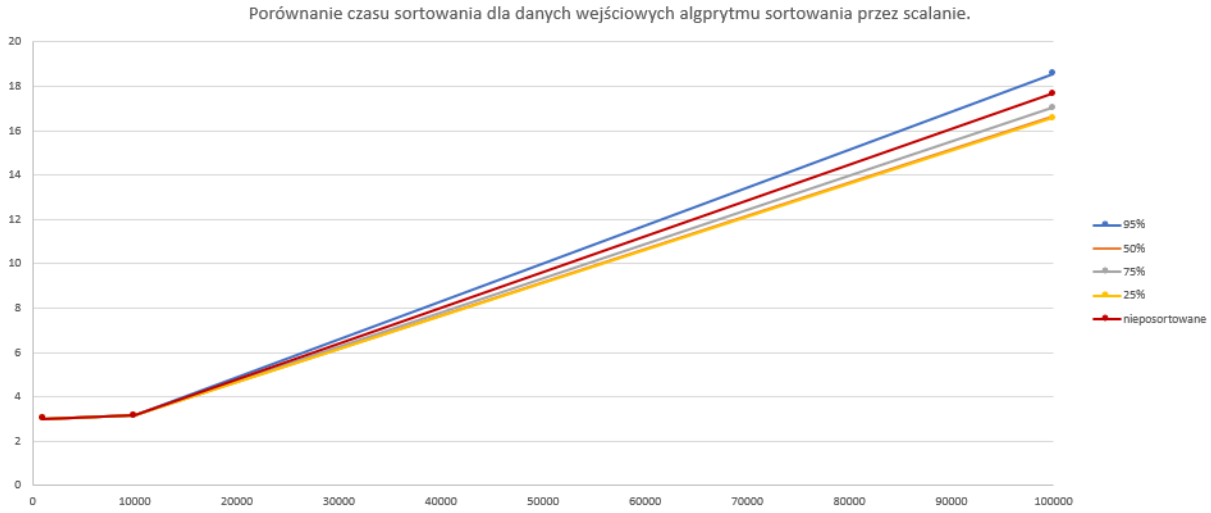
3.2 Wykresy



Z wykresów można zauważyć ze wartośći w pewnym sensie się stabilizują przez logarytm, ale widać że czas sortowania posortowanego wektora w 95% jest dłóższy przez to że jest to wartość

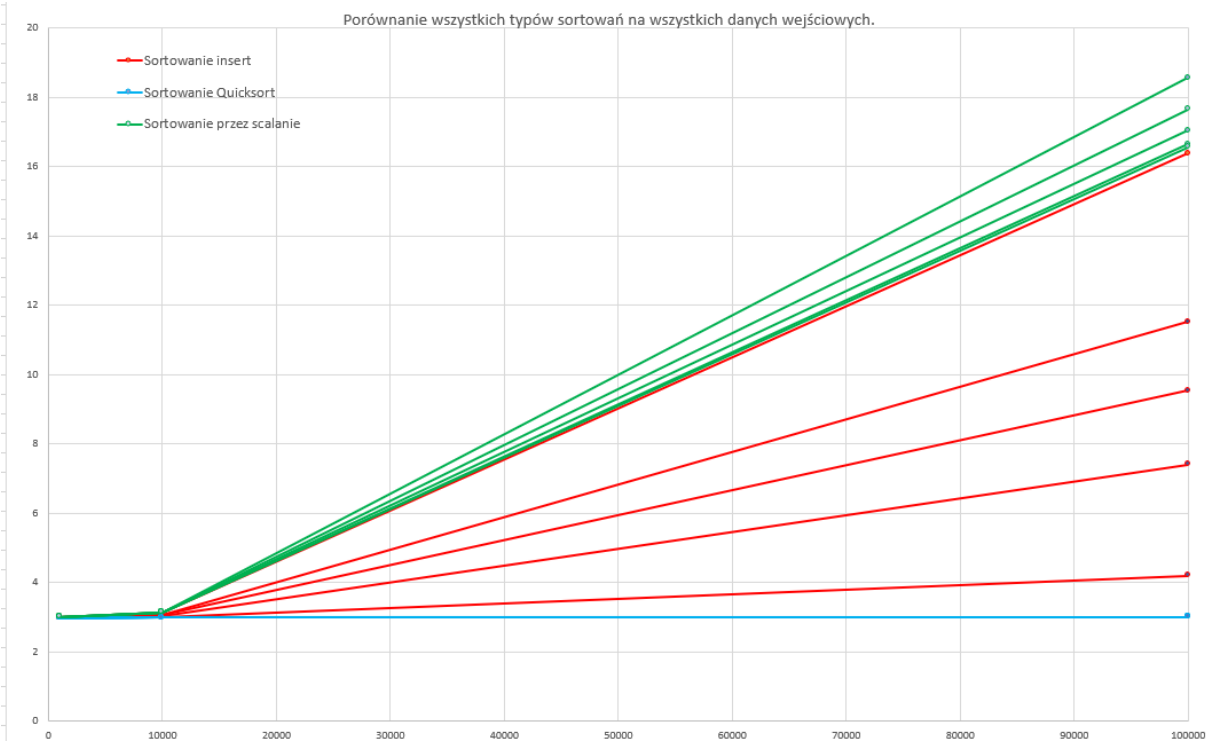


Można zauważyć, że jest to swego rodzaju zależność liniowa między następnymi sortowaniami.



Mimo, że jest zachowana pewna liniowośc to jest też tu podobienstwo do quickSortu, bo to wektor posortowany w 95% najdłużej się sortuje. (Uwaga wartości zostały minimalnie zwiększone bo exel nie wypisywał nieposortowanych 25%,75% i 50% z powodu ze były prawie identyczne, na wykresie były widoczne tylko jedna linia)

4. Wykres porównawczy



Jak łatwo zauważyć najdłużej sortuje algorytm przez scalanie, a najszybciej qiuckSort. Przez scalanie mimo, że ma podobny logarytm do quickSortu jest zdecydowanie dłuższe, i jak widać jest to dość rażąca przewaga bo ostatecznie nawet 9 razy szybciej/wolniej. Insert ma bardzo rozstrzelony czas sortowania i w przypadku małej wielkości wektora sortuje wolniej niż quickSort, ale szybciej niż przez scalanie.

5. Wnioski

Algorytm quickSort jest najbardziej optymalny ze względu na wykorzystanie ruchomego punktu rozdzielającego tablicę na dwie podtablice (Pivot). W przypadku sortowania przez scalanie punkt ten jest odgórnie ustalony i rozdziela tablicę równo w połowie, dlatego jest on znacznie dłuższy.

Wykresy czasowe quickSortu i sortowania przez scalanie jest do siebie podobne jeśli chodzi o unormowaniu się czasów w przypadku 25%,50%,75% i nieposortowanego wektora, a czas wektora posortowanego w 95% jest największy w obu algorytmach co jest spowodowane najbardziej pesymistycznym przypadkiem , czyli sortowaniu posortowanego wektora.

Sortowanie szybkie ze względu na szybkość i wydajność jest najlepszą opcja, w tym wypadku.