Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 1

Autor: Stanisław Słowik 249030

Prowadzący: Dr inż. Łukasz Jeleń

Termin zajęć Czwartek 9.15 – 11.00

Data: 31.03.2020

Opis projektu

Projekt polegał na analizie trzech algorytmów sortowania i sprawdzenia ich wydajności w zależności od wielkości tablic i procencie aktualnego posortowania tablic.
Wykorzystane będą trzy algorytmy:
- QuickSort – sortowanie szybkie
- MergeSort – sortowanie przez scalanie
- HeapSort – sortowanie przez kopcowanie

Każde sortowanie było przeprowadzane na 100 tablicach o rozmiarach:
10 000, 50 000, 100 000, 500 000, 1 000 000,
a ich stopień początkowego posortowania:
0%, 25%, 50%, 75%, 95%, 99%, 99,7%,
oraz jedna tablica, posortowana w 100% lecz w odwrotnej kolejności.

Quicksort- algorytm ten opiera się na zasadzie „dziel i zwyciężaj” w zależności od wybranego pivota (może być on elementem środkowym, pierwszym, ostatnim, losowym lub wybranym według jakiegoś innego schematu korzystnego dla naszych danych). Elementy nie większe od niego przenoszone są na lewą stronę a mniejsze na prawo od tej wartości. Powstają dwie nowe tablice które ponownie sortujemy według tego samego schematu, powtarzamy tą czynność dopóki nie uzyskamy pojedynczych posortowanych elementów.

Złożoność obliczeniowa w typowym i najlepszym przypadku: O(n log n)
Złożoność obliczeniowa w najgorszym przypadku: $O(n^2)$

MergeSort- sortowanie przez scalanie także wykorzystuje metodę „dziel i zwyciężaj”. Dzieli on dane na dwie równe części do czasu gdy pozostaną same podzbiory jednoelementowe, następnie łączy podciągi w jeden posortowany.

Złożoność obliczeniowa za każdym razem wynosi: O(n log n).

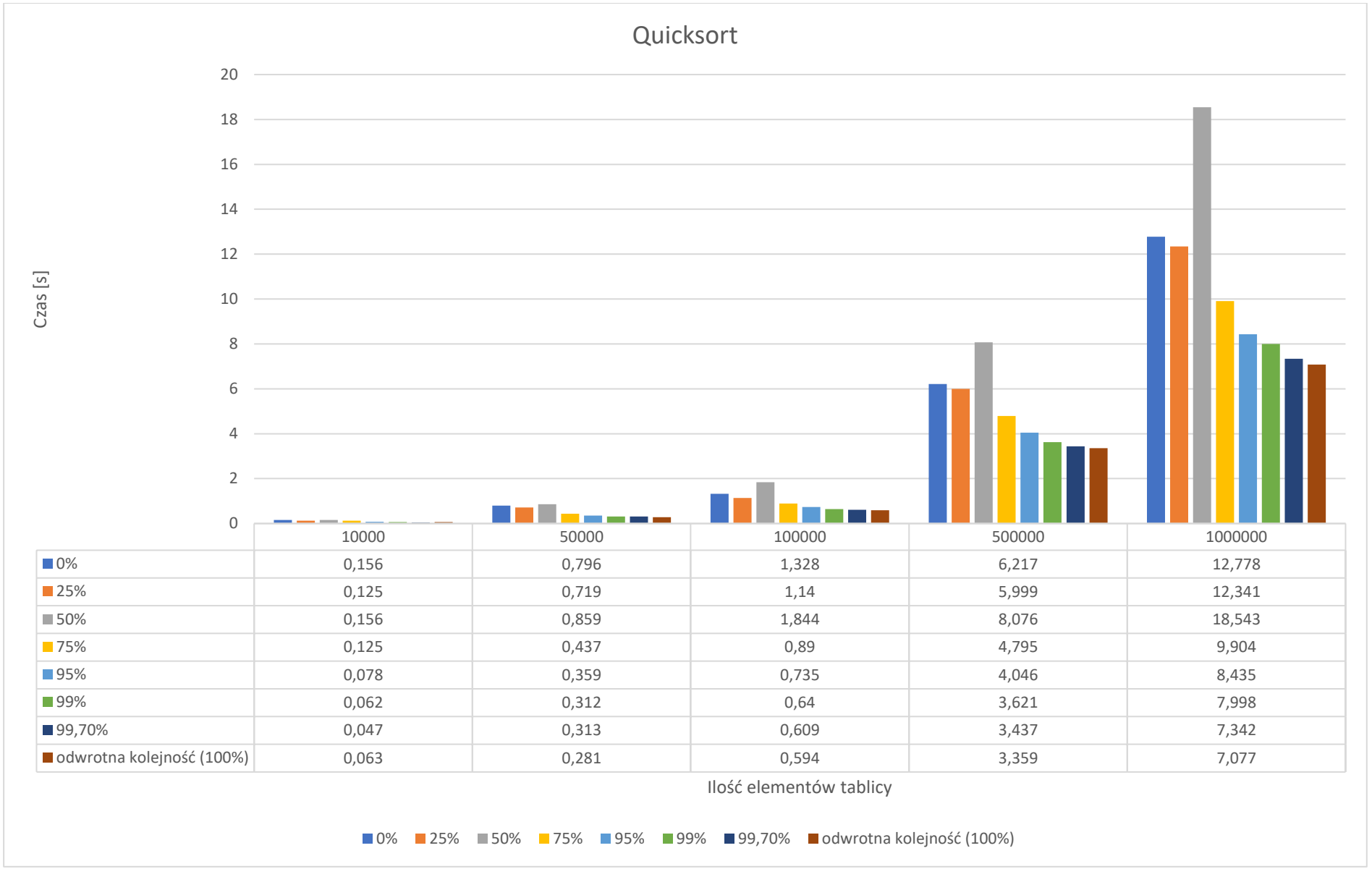
HeapSort – sortowanie przez kopcowanie, używa on kolejki priorytetowej w postaci kopca zupełnego(każdy węzeł nadrzędny jest większy lub równy węzłom potomnym). Po każdym dołączeniu nowego elementu musimy sprawdzić czy są spełnione warunki i jak trzeba dokonujemy zamian węzłów na ścieżce wiodącej od dodanego węzła do korzenia.

Złożoność obliczeniowa w każdym przypadku wynosi: O(n log n)

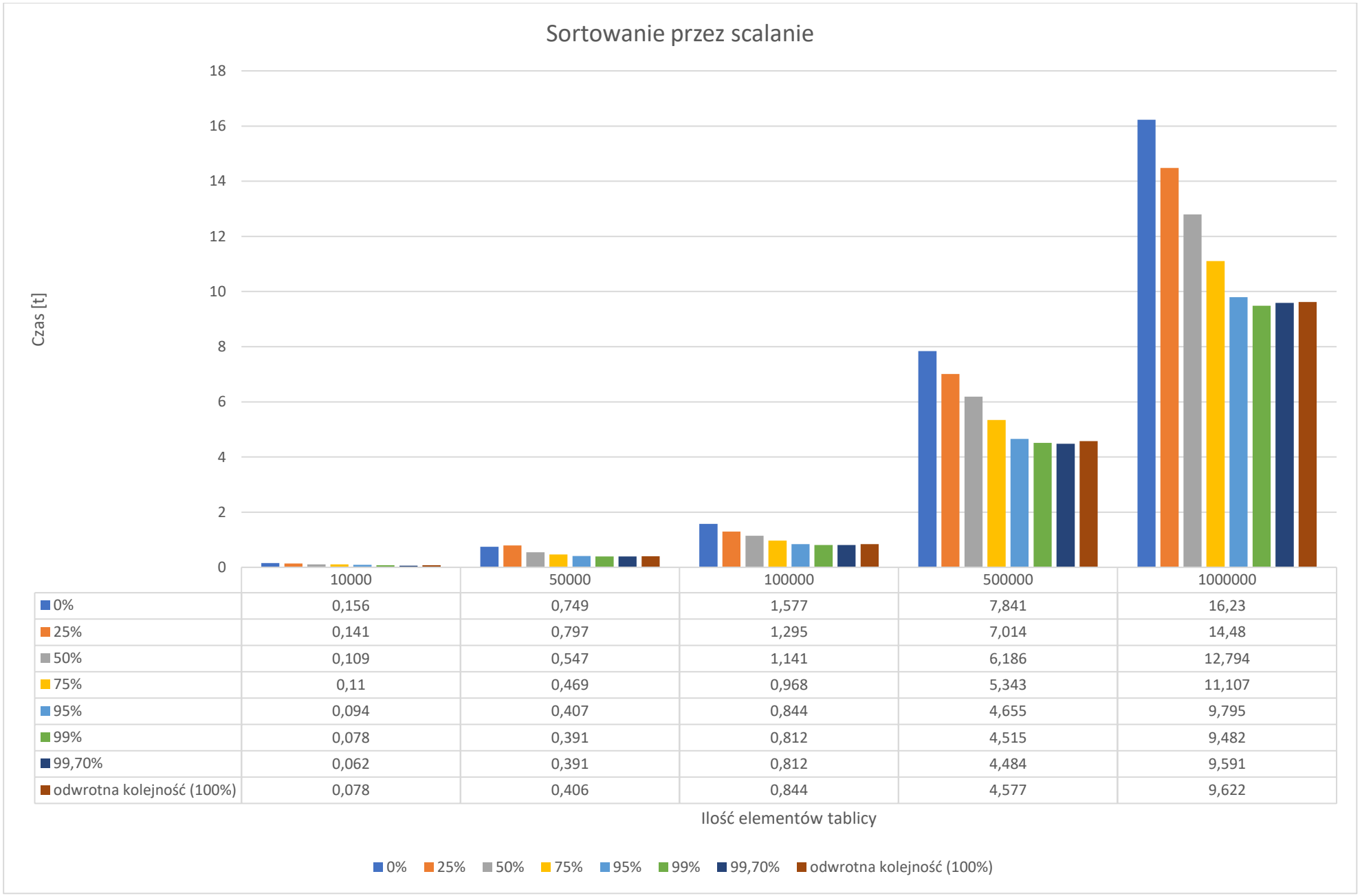
Przebieg eksperymentów

Wykonanie sortowań na wszystkich tablicach zajęło niecałe 20 minut. Pierwsze co można zauważyć to fakt, że dane wcześniej posortowane wyraźnie przyspieszają działanie algorytmu a posortowanie ich w odwrotnej kolejności nie ma negatywnego wpływu na czas działania. Od zasady jedynie odróżnia się jedynie algorytm Quicksort dla przypadku wcześniejszego posortowania 50% danych. Czas jego wykonania jest znacznie wydłużony prawdopodobnie poprzez sposób wybierania pivota co może świadczyć o niestabilność tego algorytmu.

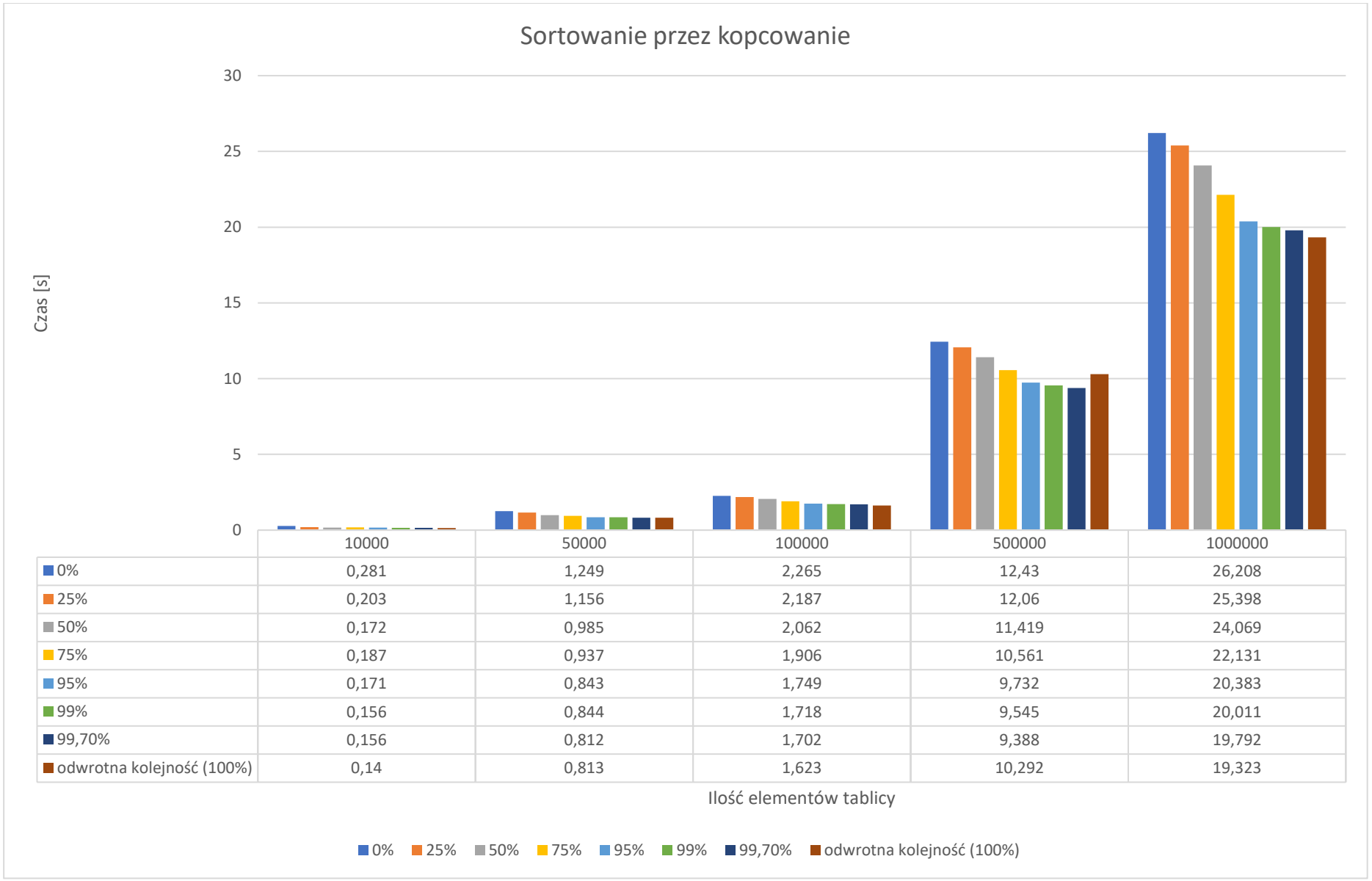
Wykresy



	10000	50000	100000	500000	1000000
0%	0,156	0,796	1,328	6,217	12,778
25%	0,125	0,719	1,14	5,999	12,341
50%	0,156	0,859	1,844	8,076	18,543
75%	0,125	0,437	0,89	4,795	9,904
95%	0,078	0,359	0,735	4,046	8,435
99%	0,062	0,312	0,64	3,621	7,998
99,70%	0,047	0,313	0,609	3,437	7,342
odwrotna kolejność (100%)	0,063	0,281	0,594	3,359	7,077



	10000	50000	100000	500000	1000000
0%	0,156	0,749	1,577	7,841	16,23
25%	0,141	0,797	1,295	7,014	14,48
50%	0,109	0,547	1,141	6,186	12,794
75%	0,11	0,469	0,968	5,343	11,107
95%	0,094	0,407	0,844	4,655	9,795
99%	0,078	0,391	0,812	4,515	9,482
99,70%	0,062	0,391	0,812	4,484	9,591
odwrotna kolejność (100%)	0,078	0,406	0,844	4,577	9,622



	10000	50000	100000	500000	1000000
0%	0,281	1,249	2,265	12,43	26,208
25%	0,203	1,156	2,187	12,06	25,398
50%	0,172	0,985	2,062	11,419	24,069
75%	0,187	0,937	1,906	10,561	22,131
95%	0,171	0,843	1,749	9,732	20,383
99%	0,156	0,844	1,718	9,545	20,011
99,70%	0,156	0,812	1,702	9,388	19,792
odwrotna kolejność (100%)	0,14	0,813	1,623	10,292	19,323

Podsumowanie i wnioski

- QuickSort jest ogólnie szybciej sortującym algorytmem od sortowania przez kopcowanie i scalanie
- Posortowanie pierwszych elementów tablicy wyraźnie skraca czas wykonywania wszytkich algorytmów
- Sortowanie przez kopcowanie jest wyraźnie najwolniejszym z 3 wybranych algorytmów, oraz posortowanie wyrazów początkowych ma najmniejszy wpływ na jego szybkość.
- Działanie algorytmów jest podobne do zakładanego z czego można wnioskować, że zostały one poprawnie zaimplementowane
- Należy uważać na to jak algorytmy zachowują się dla różnych danych, ponieważ może uwydatnić się niestabilność niektórych z nich – QuickSort ze znacznie wydłużonym, czasem dla przypadku 50% początkowych danych posortowane.

Literatura

- Cormen T., Leiserson C.E., Rivest R.L., Stein C., Wprowadzenie do algorytmów, WNT
- QuickSort- https://pl.wikipedia.org/wiki/Sortowanie_szybkie
- Sortowanie przez scalanie- https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
- Sortowanie przez kopcowanie- https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
- Sortowanie przez kopcowanie, Quicksort - https://www.geeksforgeeks.org/know-your-sorting-algorithm-set-2-introsort-cs-sorting-weapon/
- QuickSort- https://www.geeksforgeeks.org/quick-sort/
- Sortowanie przez scalanie- https://stackoverflow.com/questions/12030683/implementing-merge-sort-in-c
- C++ Primer Stanley B. Lippman, Josée Lajoie, Barbara E. Moo