25.03.2020

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 1

Autor: Dominik Brzezina 249206

Grupa: PT 9:15-11:00
Mgr. inż. Marta Emirsajłow

1 Wprowadzenie

Celem projektu było zbadanie 3 wybranych algorytmów sortowania: sortowanie szybkie (quicksort), sortowanie przez scalanie (merge sort) oraz sortowanie introspektywne (introsort), ich implementacja w języku C++ oraz przeprowadzenie testów efektywności. Testy efektywności zostały wykonane dla 100 tablic o rozmiarach:

- 10000
- 50000
- 100000
- 500000
- 1000000

dla następujących przypadków:

- wszystkie elementy tablicy losowe
- 25 %, 50%, 75%, 95%, 99%, 99,7% początkowych elementów tablicy jest już posortowanych
- tablica posortowana jest malejąco

2 Opis algorytmów

2.1 Sortowanie szybkie

W algorytmie Quicsort oryginalna tablica jest dzielona na dwie podtablice, z których pierwsza zawiera elementy mniejsze lub równe od elementów osiowego (zwanego również piwotem, granicą). W drugiej tablicy znajdują się elementy większe bądź równe od granicy. Następnie otrzymane tablice są dzielone w identyczny sposób jak powyżej, aż do uzyskania tablic jednoelementowych, które nie wymagają sortowania. Quicksort jest typowym algorytmem rekurencyjnym.
Złożoność obliczeniowa:

- Dla przypadku średniego: $O(n \log n)$
- Dla przypadku najgorszego: $O(n^2)$

2.2 Sortowanie przez scalanie

Algorytm sortowania przez scalanie (ang. mergesort) również dzieli tablice na dwie podtablice, otrzymane podtablice na kolejne itd. aż do uzyskania tablic jednoelementowych. Następnie scala je ze sobą w taki sposób, aby powstała posortowana tablica.
Złożoność obliczeniowa:

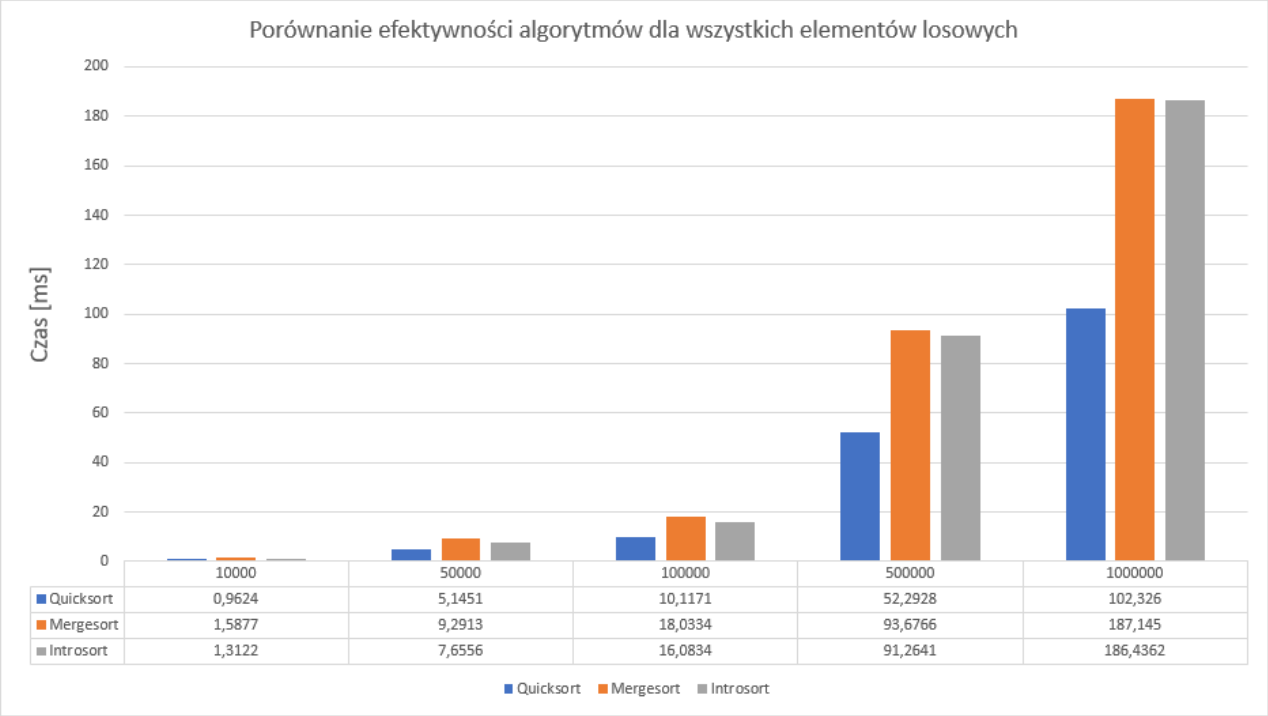
- Dla przypadku średniego: $O(n \log n)$
- Dla przypadku najgorszego: $O(n \log n)$

2.3 Sortowanie introspektywne

Sortowanie introspektywne jest połączeniem sortowania szybkiego i sortowania przez kopcowanie. Pozwala wyeliminować złożoność obliczeniową dla najgorszego przypadku sortowania szybkiego (gdy element osiowy-piwot ustawiony jest na elemencie najmniejszym lub największym tablicy, czego konsekwencją jest nierównomierny podział tablicy).
Złożoność obliczeniowa:

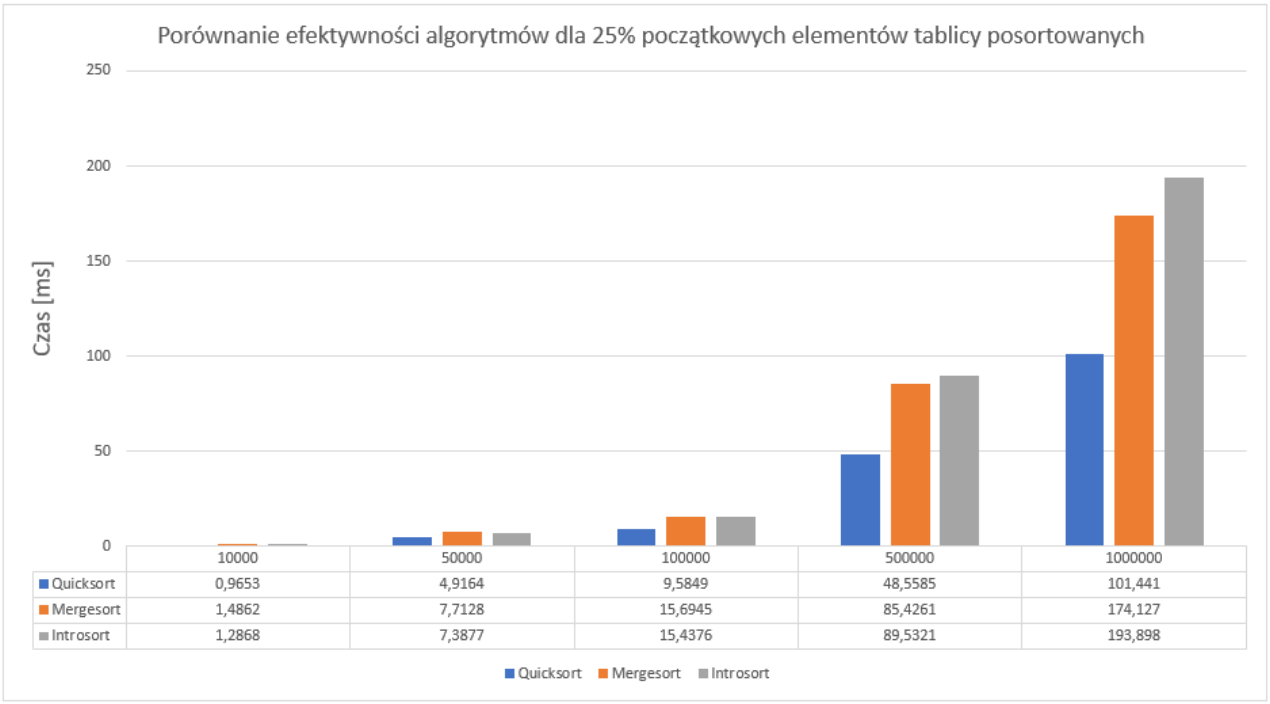
- Dla przypadku średniego: $O(n \log n)$
- Dla przypadku najgorszego: $O(n \log n)$

3 Wyniki testów



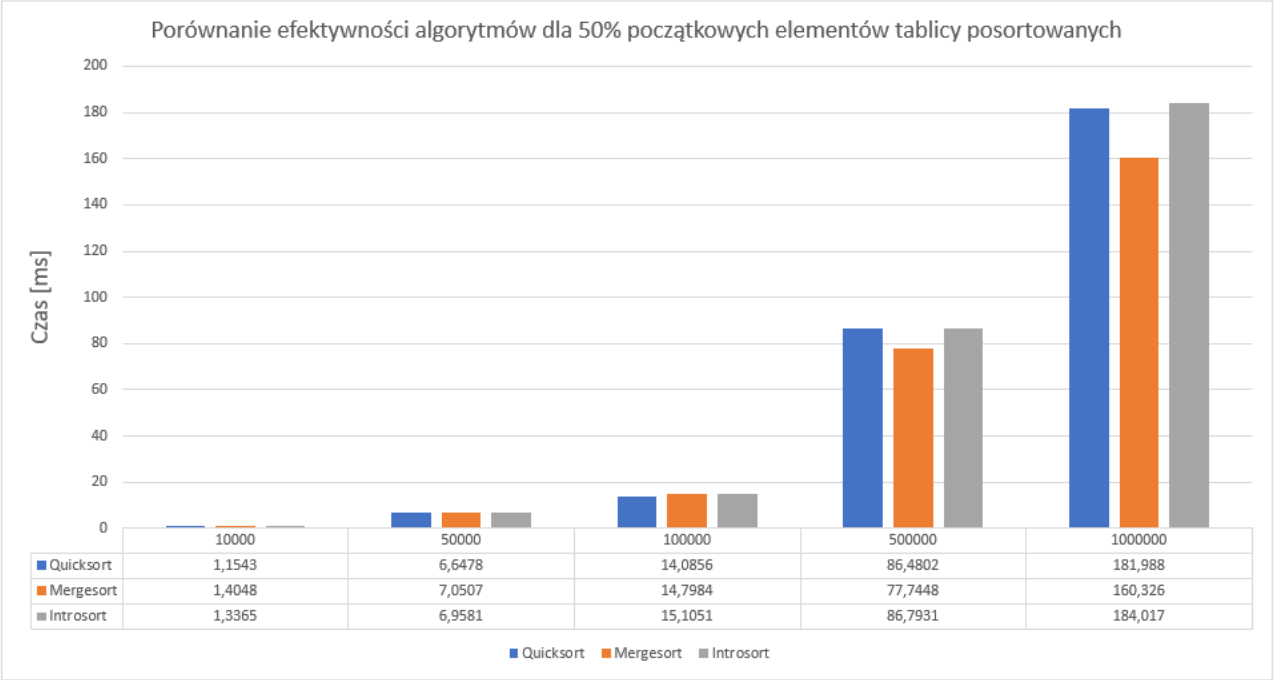
	10000	50000	100000	500000	1000000
Quicksort	0,9624	5,1451	10,1171	52,2928	102,326
Mergesort	1,5877	9,2913	18,0334	93,6766	187,145
Introsort	1,3122	7,6556	16,0834	91,2641	186,4362

Rysunek 1: Porównanie efektywności algorytmów dla wszystkich elementów losowych



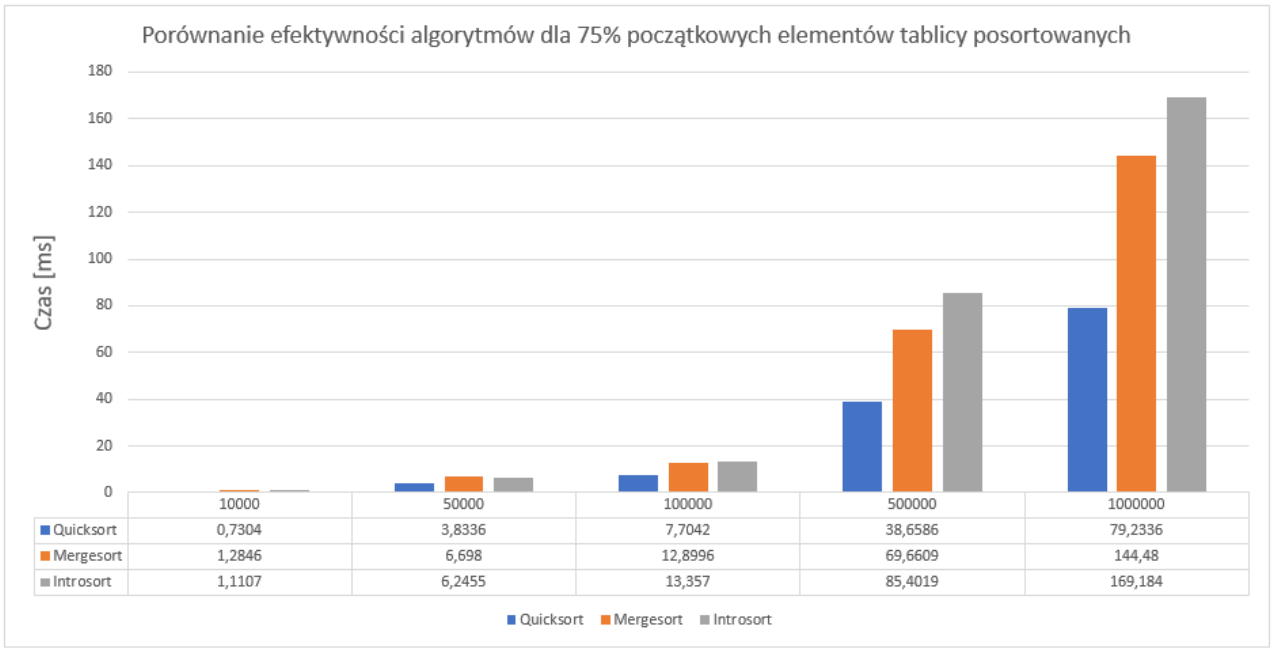
	10000	50000	100000	500000	1000000
Quicksort	0,9653	4,9164	9,5849	48,5585	101,441
Mergesort	1,4862	7,7128	15,6945	85,4261	174,127
Introsort	1,2868	7,3877	15,4376	89,5321	193,898

Rysunek 2: Porównanie efektywności algorytmów dla 25 % początkowych elementów tablicy posortowanych



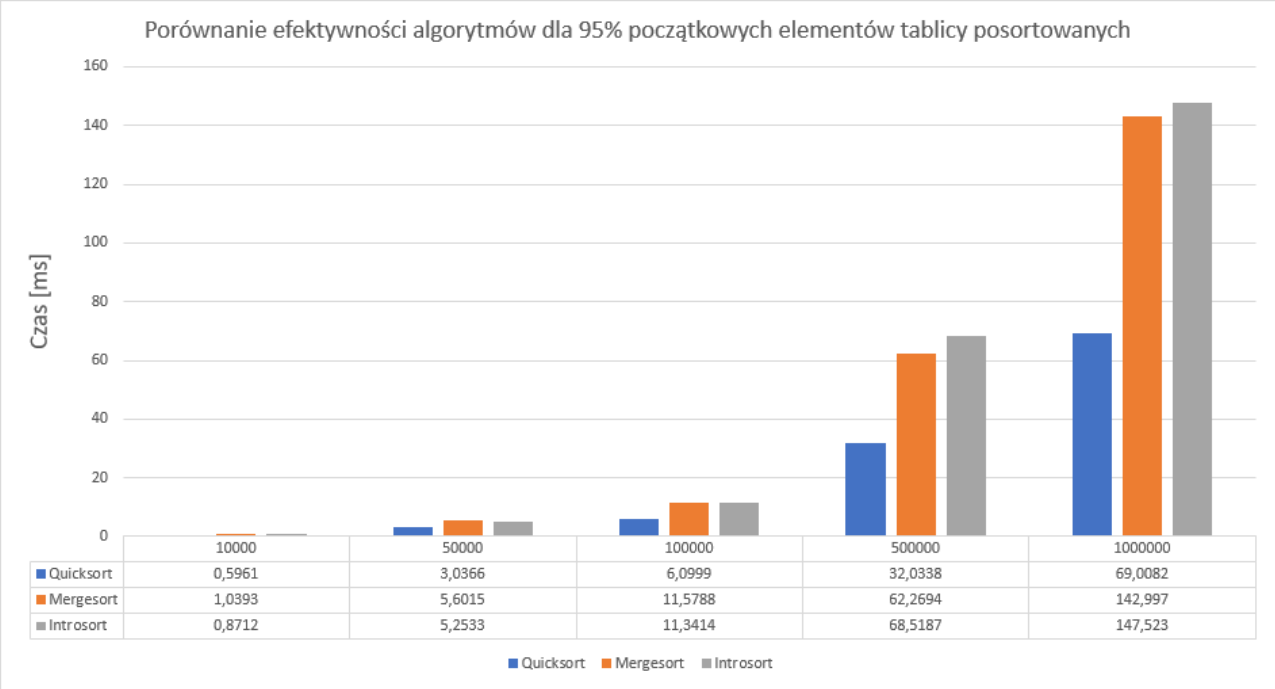
	10000	50000	100000	500000	1000000
Quicksort	1,1543	6,6478	14,0856	86,4802	181,988
Mergesort	1,4048	7,0507	14,7984	77,7448	160,326
Introsort	1,3365	6,9581	15,1051	86,7931	184,017

Rysunek 3: Porównanie efektywności algorytmów dla 50 % początkowych elementów tablicy posortowanych



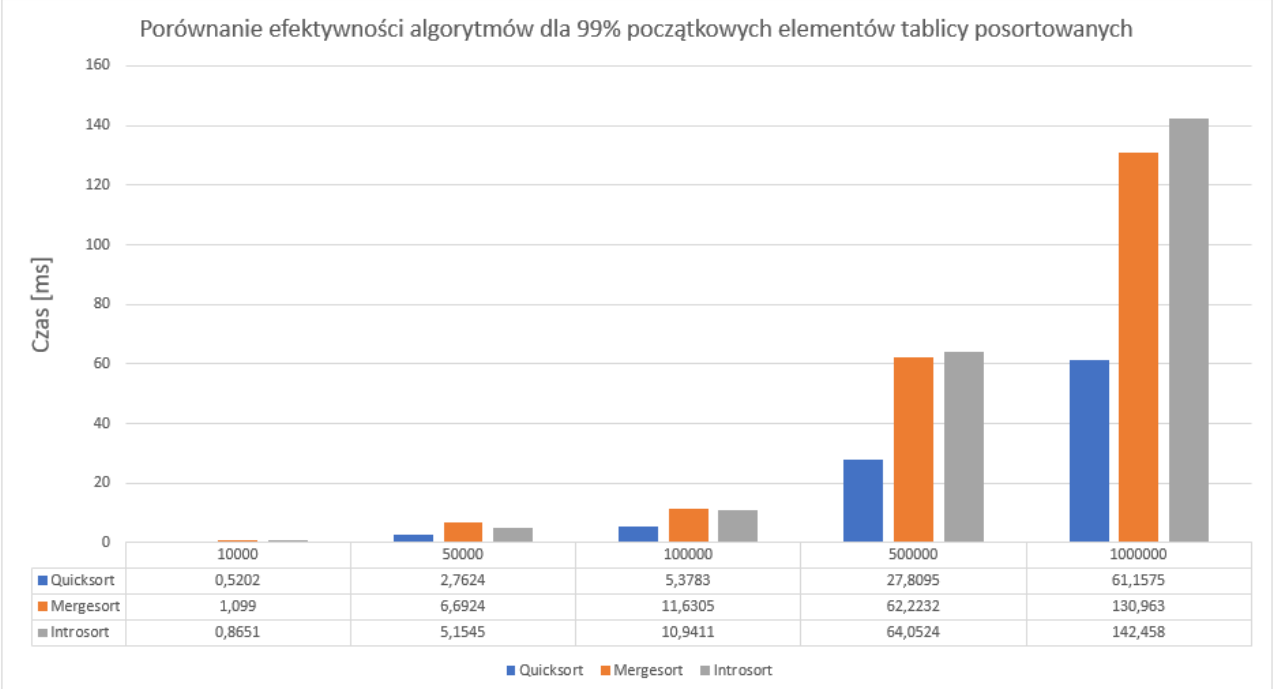
	10000	50000	100000	500000	1000000
Quicksort	0,7304	3,8336	7,7042	38,6586	79,2336
Mergesort	1,2846	6,698	12,8996	69,6609	144,48
Introsort	1,1107	6,2455	13,357	85,4019	169,184

Rysunek 4: Porównanie efektywności algorytmów dla 75 % początkowych elementów tablicy posortowanych



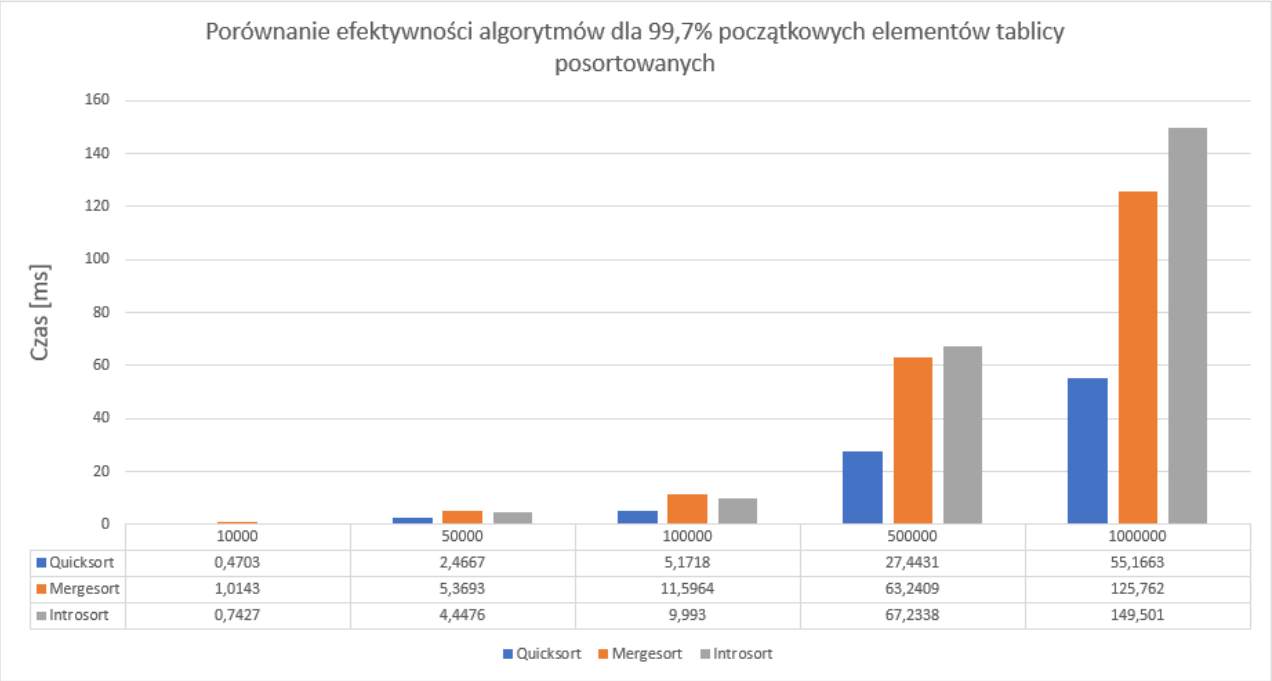
	10000	50000	100000	500000	1000000
Quicksort	0,5961	3,0366	6,0999	32,0338	69,0082
Mergesort	1,0393	5,6015	11,5788	62,2694	142,997
Introsort	0,8712	5,2533	11,3414	68,5187	147,523

Rysunek 5: Porównanie efektywności algorytmów dla 95 % początkowych elementów tablicy posortowanych



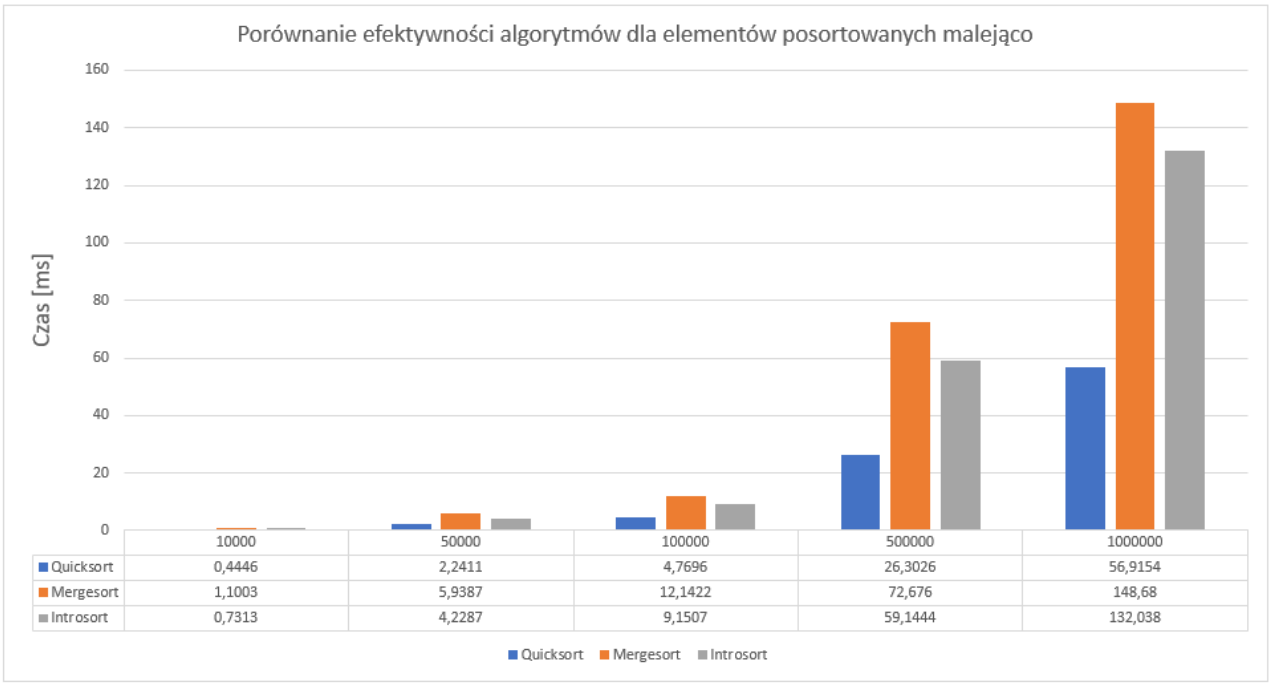
	10000	50000	100000	500000	1000000
Quicksort	0,5202	2,7624	5,3783	27,8095	61,1575
Mergesort	1,099	6,6924	11,6305	62,2232	130,963
Introsort	0,8651	5,1545	10,9411	64,0524	142,458

Rysunek 6: Porównanie efektywności algorytmów dla 99 % początkowych elementów tablicy posortowanych



	10000	50000	100000	500000	1000000
Quicksort	0,4703	2,4667	5,1718	27,4431	55,1663
Mergesort	1,0143	5,3693	11,5964	63,2409	125,762
Introsort	0,7427	4,4476	9,993	67,2338	149,501

Rysunek 7: Porównanie efektywności algorytmów dla 99,7 % początkowych elementów tablicy posortowanych



	10000	50000	100000	500000	1000000
Quicksort	0,4446	2,2411	4,7696	26,3026	56,9154
Mergesort	1,1003	5,9387	12,1422	72,676	148,68
Introsort	0,7313	4,2287	9,1507	59,1444	132,038

Rysunek 8: Porównanie efektywności algorytmów dla tablicy uporządkowanej malejąco

4 Podusmowanie

- Zgodnie z oczekiwaniami dla przypadków z wszystkimi losowymi elementami oraz tablicami posortowanymi malejąco najwolniejszy okazał się Mergesort. Zbliża się on pod względem efektywności tylko w przypadku, gdy część tablicy jest posortowana - dla tablic posortowanych w 50 % okazał się jedyny raz najszybszy spośród wszystkich algorytmów.
- Nieoczekiwany rezultat otrzymano dla algorytmów Quicksort oraz Introsort. W teorii, najszybciej sortować powinien algorytm Introsort, jako że posiada zabezpieczenie przed złożonością obliczeniową dla najgorszego przypadku występującej dla Quicksorta. W czasie testów okazało się, że algorytm Quicksort sortuje znacznie szybciej każdy rodzaj oraz rozmiar tablicy. Wpływ ma na to zapewne prosty i krótki kod implementujący algorytm szybkiego sortowania, a przedewszystkim umiejscowienie piwota w środkowej części tablicy, dzięki czemu nie występuje w nim złożoność obliczeniowa dla najgorszego przypadku.

5 Literatura

- A. Drozdek (2004) "C++ Algorytmy i struktury danych" Wydawnictwo Helion, Gliwice
- Wikipedia "Sortowanie introspektywne" https://pl.wikipedia.org/wiki/Sortowanie_introspektywne