13.04.2020r.

Paweł Troszczyński 248925

Prowadzący zajęcia: mgr inż. Marcin Ochman

Termin zajęć: wtorek 15:15-16:55

Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Algorytmy sortowania

1. Wstęp

Celem projektu jest zaimplementowanie i zbadanie trzech algorytmów sortowania pod względem ich czasu działania i złożoności obliczeniowej. Badanymi algorytmami są sortowanie przez scalanie, sortowanie szybkie oraz sortowanie introspektywne.

2. Opis badanych algorytmów

a) sortowanie przez scalanie:

Jest to rekurencyjny algorytm wykorzysujący zasadę „dziel i zwyciężaj". Dzieli on zestaw otrzymanych danych na dwie równe części (lub różniące się o jeden w przypadku nieparzystych zestawów). Operacja ta powtarzana jest aż do momentu, gdy zostanie tylko jeden element w części, która powstała przez dzielenie. Następnie algorytm łączy ze sobą(„scala") dwie otrzymane części posortowane (część zawierająca jeden element jest posortowana) w jedną posortowaną część aż do momentu, gdy otrzyma ponownie wszystkie otrzymane dane w jednej. W ten sposób algorytm ten sortuje otrzymywane dane. Wywołanie tego sortowania powoduje wywołanie $\log_2 \mathrm{n}$ rekurencji, gdzie 'n' to liczba elementów, które mają zostać posortowane. Dodatkowo algorytm ten wywołuje 'n' razy pewna stała scaleń. Daje to złożoność obliczeniową tego algorytmu wynoszącą $\mathrm{O}(\mathrm{n}\log\mathrm{n})$. Zarówno w przypadku średnim jak i najgorszym złożoność obliczeniowa tego algorytmu jest taka sama, ponieważ zmianie ulegać może tylko stała, zaś potęga 'n' pozostaje niezmienna.

b) sortowanie szybkie:

Algorytm ten polega na wyznaczeniu tzw. „pivot'u", czyli elementu rozdzielającego, który będzie dzielił tablice na dwa fragmenty. Następnie wszystkie elementy są tak przenoszone, aby elementy nie większe od „pivot'u" były po jego lewej stronie, zaś te nie mniejsze po jego prawej stronie. Algorytm ten powtarza ten proces aż do uzyskania tablic zawierających jeden element, ponieważ takie są już posortowane. Liczba rekurencji jest zależna od przyjmowanego przez algorytm „pivot'u". Daje to w średnim przypadku złożoność obliczeniową $\mathrm{O}(\mathrm{n}\log\mathrm{n})$. Jednak w najgorszym przypadku, gdy wybieranym „pivot'em" będzie cały czas wartość największa lub najmniejsza to złożoność obliczeniowa tego algorytmu wzrośnie do $\mathrm{O}(\mathrm{n}^2)$.

c) sortowanie introspektywne:

Jest to sortowanie hybrydowe. Jego głównym celem jest pozbycie się problemu złożoności obliczeniowej $\mathrm{O}(\mathrm{n}^2)$ w najgorszym przypadku algorytmu sortowania szybkiego. Algorytm ten polega na połączeniu trzech algorytmów sortowania: szybkie, przez kopcowanie i przez wstawianie. Sortowanie przez wstawianie posiada złożoność obliczeniową $\mathrm{O}(\mathrm{n}^2)$ jednak dla małych tablic (eksperymentalnie wyznaczono, że dla tablic o 9 lub mniejszej liczbie elementów) sortowanie przebiega szybciej. Dlatego też, gdy na skutek dzielenia tablicy przez sortowanie szybkie powstanie fragment o 9 lub mniej elementach, zostanie on posortowany algorytmem sortowania przez wstawianie. Sortowanie introspektywne wykorzystuje pewną stałą do określenia, który z algorytmów sortowania zostanie zastosowany: sortowanie szybkie czy przez kopcowanie. Stała ta ma wartość $2 \cdot \log_2 \mathrm{n}$ gdzie 'n' to liczba elementów do posortowania. Określa ona maksymalną głębokość wywołań rekurencyjnych. Algorytm zaczyna sortować identycznie do algorytmu sortowania szybkiego, jednak po przekroczeniu maksymalnej głębokości wywołań rekurencyjnych przełącza się na sortowanie przez kopcowanie. Powoduje to, że zarówno dla przypadku średniego jak i najgorszego złożoność obliczeniowa takiego algorytmu jest równa $\mathrm{O}(\mathrm{n}\log\mathrm{n})$.

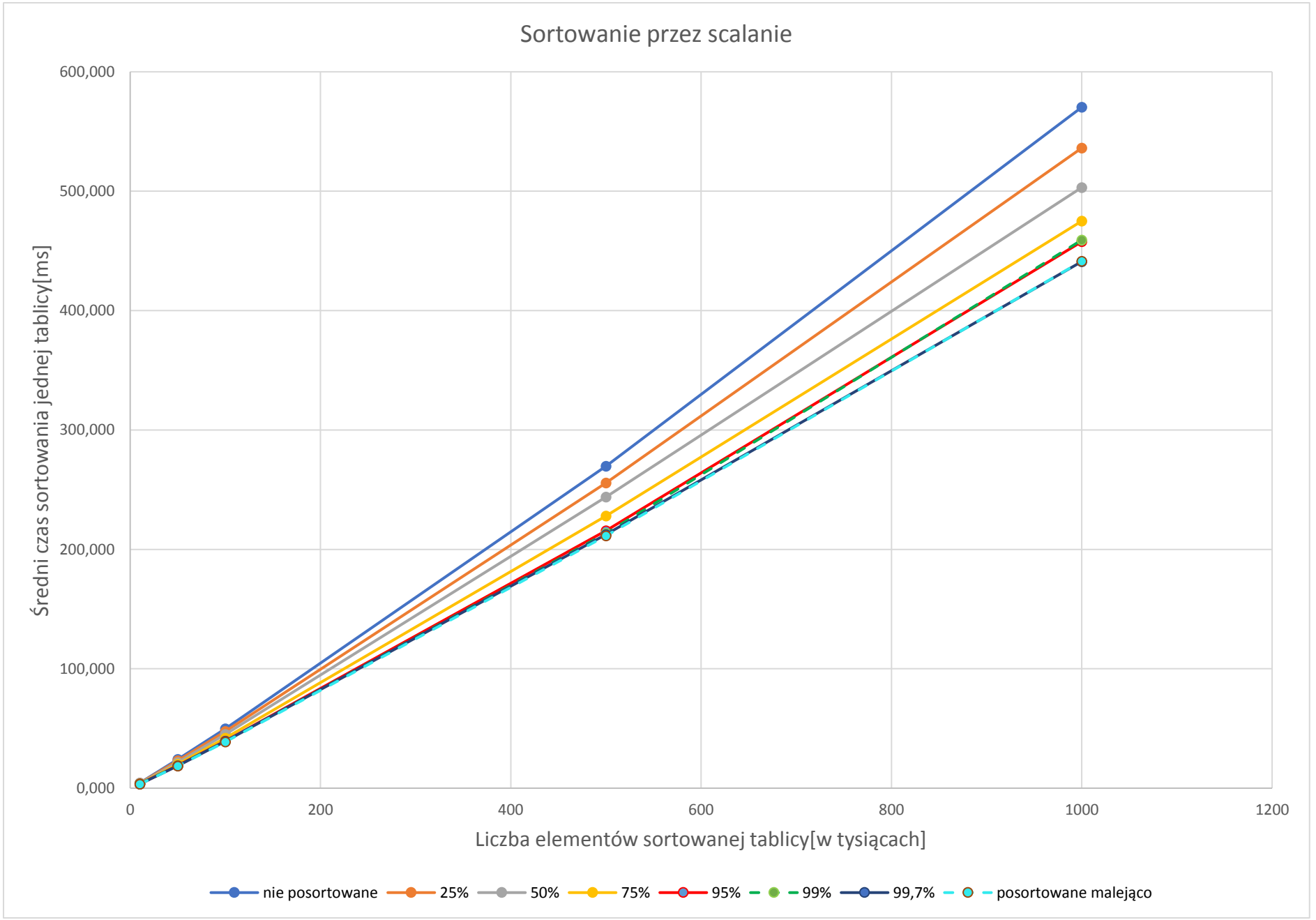
3. Przebieg eksperymentów i otrzymane wyniki

Dla każdego z trzech badanych algorytmów wygenerowano 100 tablic o rozmiarach: 10 000, 50 000, 100 000, 500 000 i 1 000 000 dla ośmiu różnych przypadków:

- gdy wszystkie elementy tablic są losowe
- gdy 25%, 50%, 75%, 95%, 99%, 99,7% początkowych elementów tablic jest posortowane
- gdy wszystkie elementy tablicy są posortowane malejąco.

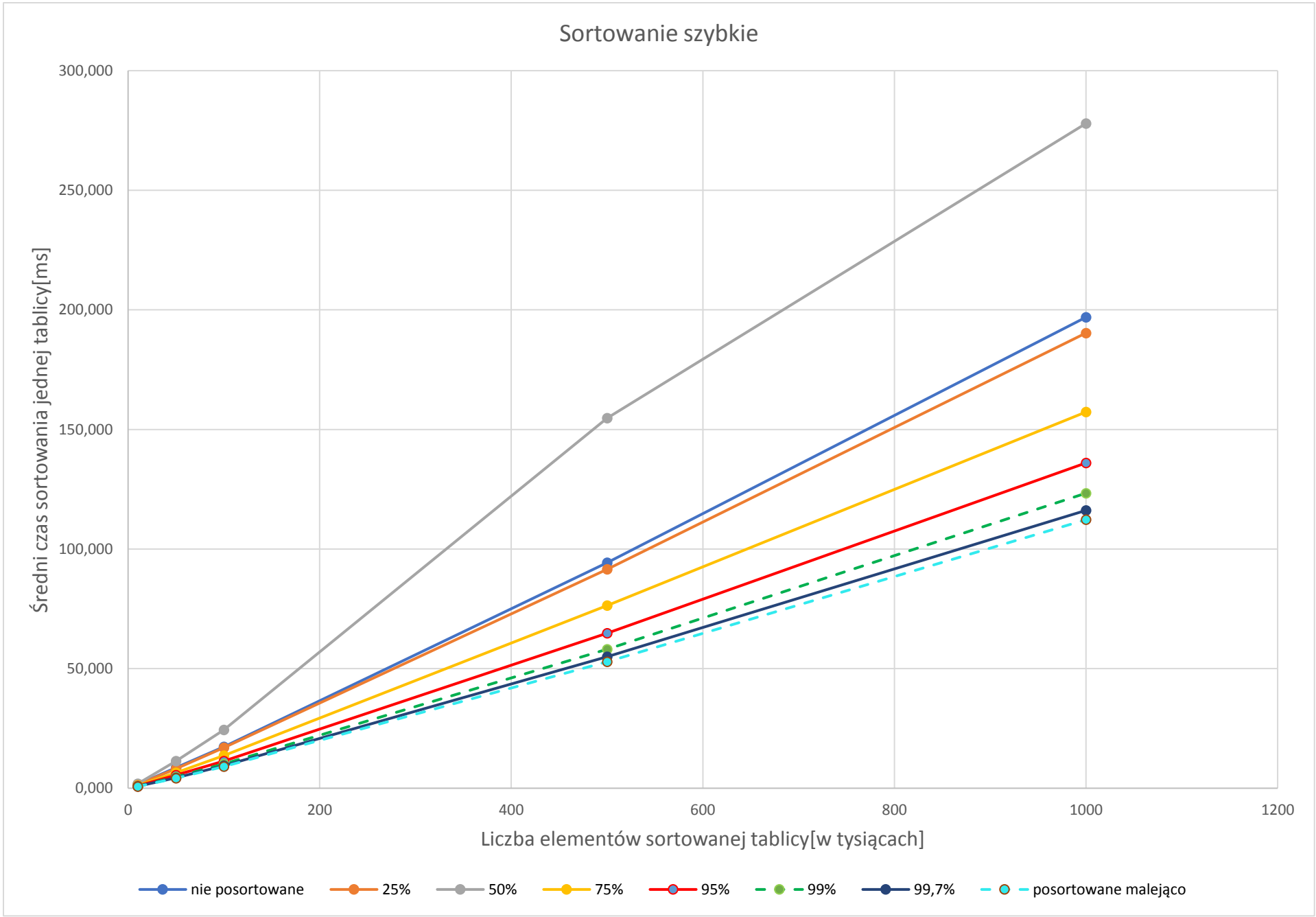
Każdy z badanych algorytmów sortuje dane w otrzymanej tablicy rosnąco. Elementem rozdzielającym w sortowaniu szybkim jest element środkowy. Wartości losowanych elementów były z zakresu od 0 do 999. Dla każdego z opisanych wyżej przypadków zmierzono czas sortowania każdej jednej tablicy. Następnie określono najdłuższy oraz najkrótszy czas potrzebny algorytmowi na posortowanie otrzymanych elementów oraz uśredniono wszystkie 100 otrzymanych wyników czasowych dla każdego z przypadku. Poniżej zostaną przedstawione wartości największe, najmniejsze i średnia czasu w jakim algorytm sortował jedną tablicę. Pozwoli to określić różnicę między najlepszym czasem sortowania a najgorszym. Średni czas natomiast pozwoli określić jakość sortowania badanych algorytmów, ponieważ im mniejszy czas średni danego sortowania tym sortowanie jest lepsze od sortowania o większym czasie średnim. Z tego też powodu wykresy poszczególnych algorytm ów sortowania ukazują czas średni sortowania jednej tablicy o zadanej liczbie elementów.

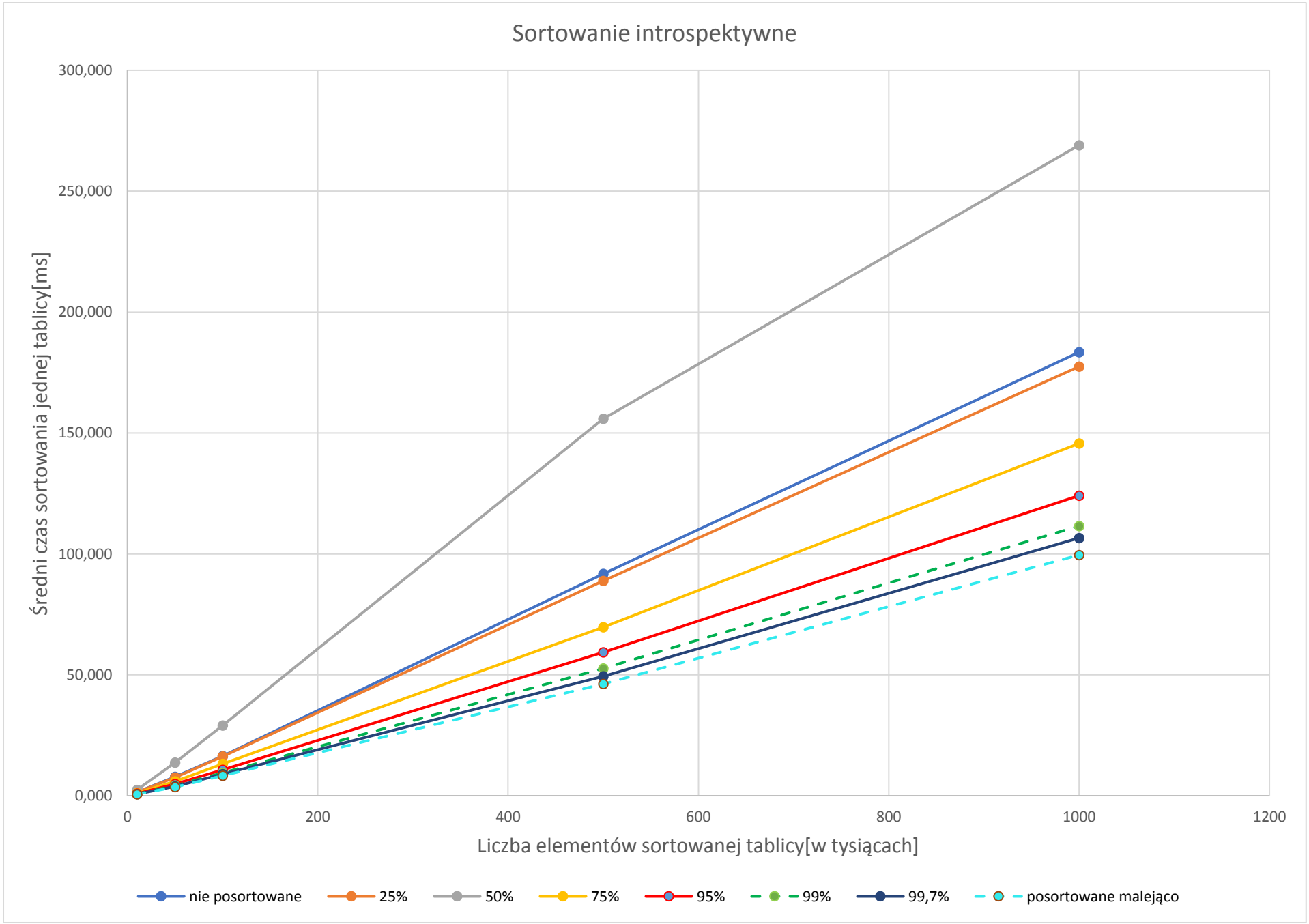
Sortowanie przez scalanie									
Liczba elementów	Czas[ms]	Stopień posortowania							
		nie posortowane	25,00%	50,00%	75,00%	95,00%	99,00%	99,70%	posortowane malejąco
10000	największy	4,610	4,918	5,231	4,128	3,912	3,793	3,779	3,589
	najmniejszy	4,113	3,918	3,722	3,477	3,303	3,333	3,279	3,086
	średni	4,290	4,123	3,982	3,705	3,504	3,517	3,431	3,330
50000	największy	28,340	24,349	21,995	20,928	19,996	19,871	20,756	20,098
	najmniejszy	23,091	22,469	20,968	19,765	18,530	18,184	18,313	18,095
	średni	24,090	22,950	21,454	20,094	18,937	18,871	18,705	18,571
100000	największy	53,194	53,150	50,455	46,406	41,093	41,266	41,393	48,450
	najmniejszy	48,608	46,494	43,933	41,244	38,832	38,248	38,396	37,733
	średni	49,815	47,551	45,070	41,911	39,570	39,079	39,288	38,769
500000	największy	284,637	260,487	266,258	251,802	229,086	236,255	215,576	214,725
	najmniejszy	267,603	253,815	240,394	225,215	213,343	211,453	211,106	209,203
	średni	269,724	255,672	243,863	227,956	215,683	213,438	212,415	211,305
1000000	największy	690,989	545,811	510,992	493,129	495,120	539,812	465,237	454,507
	najmniejszy	559,342	528,459	498,941	470,619	446,579	442,535	437,508	437,102
	średni	570,325	536,165	503,120	474,995	457,739	459,103	441,053	441,316


Sortowanie przez scalanie
600,000
500,000
400,000
300,000
200,000
100,000
0,000
Średni czas sortowania jednej tablicy[ms]
0
200
400
600
800
1000
1200
Liczba elementów sortowanej tablicy[w tysiącach]
nie posortowane
25%
50%
75%
95%
99%
99,7%
posortowane malejąco

Sortowanie szybkie									
Liczba elementów	Czas[ms]	Stopień posortowania							
		nie posortowane	25,00%	50,00%	75,00%	95,00%	99,00%	99,70%	posortowane malejąco
10000	największy	1,884	1,985	4,290	1,654	1,225	0,995	1,081	0,837
	najmniejszy	1,466	1,426	1,333	1,137	0,913	0,807	0,716	0,672
	średni	1,576	1,559	1,850	1,225	0,986	0,865	0,800	0,722
50000	największy	12,638	8,712	33,970	7,376	6,980	5,395	4,557	4,665
	najmniejszy	8,140	7,827	7,581	6,294	5,259	4,572	4,149	4,045
	średni	8,575	8,174	11,389	6,527	5,465	4,775	4,375	4,238
100000	największy	18,430	17,804	86,325	14,266	12,442	11,280	10,278	10,395
	najmniejszy	16,797	16,392	15,643	13,365	11,059	9,661	9,091	8,746
	średni	17,351	16,950	24,387	13,683	11,375	10,137	9,430	9,032
500000	największy	99,039	95,174	734,930	97,779	70,650	63,396	59,619	55,458
	najmniejszy	91,906	89,598	86,040	74,172	63,571	57,143	54,002	51,992
	średni	94,262	91,545	154,712	76,362	64,774	58,125	55,017	52,875
1000000	największy	203,302	198,562	1385,720	163,249	143,372	151,189	126,480	119,057
	najmniejszy	191,866	184,927	177,262	154,398	134,255	121,607	115,005	109,902
	średni	196,917	190,329	277,931	157,298	136,003	123,355	116,168	112,257

Sortowanie introspektywne									
Liczba elementów	Czas[ms]	Stopień posortowania							
		nie posortowane	25,00%	50,00%	75,00%	95,00%	99,00%	99,70%	posortowane malejąco
10000	największy	1,764	1,840	4,720	1,591	1,027	1,157	0,944	0,957
	najmniejszy	1,369	1,303	1,247	0,989	0,787	0,667	0,588	0,557
	średni	1,465	1,435	2,414	1,098	0,856	0,730	0,653	0,612
50000	największy	8,726	8,254	27,123	6,692	5,337	4,449	4,233	4,138
	najmniejszy	7,483	7,245	7,003	5,683	4,715	4,049	3,679	3,435
	średni	7,846	7,561	13,713	5,923	4,887	4,233	3,877	3,611
100000	największy	17,926	26,301	66,528	26,341	17,449	21,557	17,809	17,288
	najmniejszy	15,766	14,955	14,698	12,081	10,218	8,876	8,178	7,633
	średni	16,424	16,212	29,018	13,149	10,755	9,489	8,875	8,241
500000	największy	110,387	108,116	331,360	72,660	63,796	54,663	52,544	51,526
	najmniejszy	86,518	83,392	79,289	67,616	57,606	51,849	48,032	45,071
	średni	91,778	88,892	155,890	69,702	59,299	52,586	49,402	46,204
1000000	największy	211,264	183,884	677,847	171,472	127,953	113,850	121,834	118,445
	najmniejszy	179,472	171,919	166,151	143,337	123,088	110,248	104,349	97,442
	średni	183,409	177,489	268,986	145,661	124,108	111,552	106,586	99,551


Sortowanie szybkie
300,000
250,000
200,000
150,000
100,000
50,000
0,000
Średni czas sortowania jednej tablicy[ms]
0
200
400
600
800
1000
1200
Liczba elementów sortowanej tablicy[w tysiącach]
nie posortowane
25%
50%
75%
95%
99%
99,7%
posortowane malejąco


Sortowanie introspektywne
300,000
250,000
200,000
150,000
100,000
50,000
0,000
Średni czas sortowania jednej tablicy[ms]
0
200
400
600
800
1000
1200
Liczba elementów sortowanej tablicy[w tysiącach]
nie posortowane
25%
50%
75%
95%
99%
99,7%
posortowane malejąco

4. Podsumowanie i wnioski

- Średnio najwolniejszym z badanych algorytmów jest algorytm sortowania przez scalanie. Średnio najszybszym natomiast algorytm sortowania introspektywnego.
- Sortowanie przez scalanie posiada najmniejszą ze wszystkich różnicę między najgorszym czasem sortowania a najmniejszym. Oznacza to, że jest to najbardziej stabilny pod względem czasu sortowania algorytm z trzech badanych. Wykorzystując ten algorytm czas sortowania będzie zawsze zbliżony do średniej czasu sortowania tego algorytmu.
- Czas sortowania tablic algorytmami szybkim oraz introspektywnym jest najgorszy dla przypadku, gdy tablica uporządkowana jest już wcześniej w 50%. Dzieje się tak, ponieważ wybieranym elementem rozdzielającym sortowania szybkiego jest element środkowy. Więc częstym przypadkiem może być sytuacja taka, że elementem rozdzielającym zostaje element największy. Średni czas w tym przypadku jest mniejszy dla algorytmu sortowania szybkiego poza tablicą o 1 000 000 elementów. Jednak wartości maksymalne jakie przyjmują oba algorytmy są dużo niższe dla algorytmu introspektywnego (z wyjątkiem tablicy 10 000 elementów). Wraz ze wzrostem liczby elementów sortowanych tablic algorytm introspektywny posiada coraz mniejszą różnice między średnimi czasami tych dwóch sortowań, aż w pewnym momencie posiada średnio czasowo lepszy wynik.
- Sortowanie introspektywne spełnia swoje założenia. W przypadkach innych niż najgorszy, dzięki wykorzystaniu połączenia sortowania szybkiego oraz sortowania przez wstawianie dla małych tablic algorytm ten osiąga najlepsze średnie czasy sortowania. W przypadku najgorszym algorytm zapobiega wysokiej złożoności obliczeniowej sortowania szybkiego(dlatego też wartości maksymalne uzyskanych czasów są dużo mniejsze niż dla samego sortowania szybkiego we wszystkich przypadkach poza tym opisanym powyżej). Algorytm ten w takim momencie wykorzystuje sortowanie przez kopcowanie, które jest wolniejsze niż sortowanie szybkie w przypadkach średnich, ale posiada zawsze złożoność obliczeniową $O(n \log n)$. Jeżeli po przekroczeniu określonej wcześniej głębokości rekurencji algorytm ten przełączy się na algorytm sortowania przez kopcowanie, zaś dla sortowania szybkiego będzie to ostatnie wywołanie rekurencji to skutkiem tego może być to, że algorytm sortowania introspektywnego zakończy sortowanie wolniej niż samo sortowanie szybkie. Ten powód może być przyczyną gorszego średniego wyniku algorytmu introspektywnego w przypadku tablic posortowanych w 50%.

Literatura:

https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
https://pl.wikipedia.org/wiki/Sortowanie_szybkie
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
https://en.cppreference.com/w/cpp/chrono
https://appdividend.com/2019/04/30/merge-sort-in-cpp-tutorial-with-example-cpp-merge-sort-program/
https://www.geeksforgeeks.org/cpp-program-for-quicksort/
https://www.geeksforgeeks.org/heap-sort/
http://www.cplusplus.com/reference/cstdlib/rand/
https://eduinf.waw.pl/inf/alg/001_search/0113.php