Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 1. Algorytmy sortujące

Michał Maćkowiak 249464

23.03.2020

Prowadzący:

dr hab. inż. Andrzej Rusiecki

Zajęcia:

Wtorek godz. 13.15

1. Algorytmy

1.1. Sortowanie przez scalanie (Merge sort)

Sortowanie to jest algorytmem typu „Divide and conquer", czyli dziel i zwyciężaj. By dokonać sortowania należy rekurencyjnie dzielić tablice na połowy, aż do uzyskania jednoelementowych pod problemów. Następnie scalać je w posortowanej kolejności, aż powstanie posortowana tablica wejściowa.

Złożoność obliczeniowa Merge sort'a:

- w najgorszym przypadku wynosi $\mathcal{O}(n\ logn)$,
- w średnim przypadku wynosi $\mathcal{O}(n\ logn)$.

Można zauważyć, że algorytmie nie występuje żaden przypadek skrajny, różniący się od pozostałych.

Algorytm ten jest stabilny to znaczy, że w przypadku napotkania tych samych elementów, występujących po sobie, nie zostaną one zamieniane miejscami.

1.2. Sortowanie szybkie (Quicksort)

Algorytm wykorzystuje technikę „Divide and conquer". Na początku sortowania wybierany jest pivot (jeden z elementów tablicy), który dzieli tablice na dwie części. Zgodnie z wartością pivot'u elementy są przydzielane do dwóch nowych tablic. Do jednej wartości mniejsze od elementu rozdzielającego, w drugiej zaś elementy większe. Algorytm wywołuje się rekurencyjnie dla każdej nowo powstałej tablicy, aż do momentu gdy tablice będą jednoelementowe.

Złożoność obliczeniowa Quicksort'a:

- w najgorszym przypadku wynosi $\mathcal{O}(n^2)$,
- w średnim przypadku wynosi $\mathcal{O}(n\ logn)$.

Pesymistyczny przypadek dotyczy przypadku, gdy za pivot zostanie wybrany element największy lub najmniejszy. Sortowanie to jest niestabilne.

1.3. Sortowanie introspektywne (Introsort)

Algorytm sortowania hybrydowy. Oznacza to, że wykorzystuje kilka algorytmów sortowania. Introsort jest połączeniem algorytmu sortowania szybkiego, sortowania przez kopcowanie oraz sortowania przez wstawianie. Zasada działania algorytmu polega na wybraniu elementu rozdzielającego, czyli pivotu (jak w Quicksort). Jeżeli ilość wywołań rekursywnych tej samej funkcji (głębokość rekurencji $2 \cdot logn$) jest równa 0 to sortowanie rozpoczyna się przy pomocy sortowania przez kopcowanie. Dodatkowo jeżeli zestaw danych spradnie poniżej zadanego poziomu dalej sortowane jest przy pomocy sortowania przez wstawianie. Taka implementacja sortowania pozwala przyspieszyć jego działanie eliminując pesymistyczny przypadek sortowana szybkiego.

Złożoność obliczeniowa Introsort'u:

- w najgorszym wypadku wynosi $\mathcal{O}(n\ logn)$,
- w średnim przypadku wynosi $\mathcal{O}(n\ logn)$.

Dzięki zastosowaniu sortowania hybrydowego najgorszy przypadek sortownia szybkiego jest wyeliminowany. Sortowanie to jest niestabilne.

2. Przebieg eksperymentów

Po uruchomieniu programu tworzone jest po 100 tablic elementów losowych o rozmiarach:

- 10 000,
- 50 000,
- 100 000,
- 500 000,
- 1 000 000.

Utworzone tablice były sortowane w przypadku gdy ich część jest posortowana. Przypadki z eksperymentów to:

- wszystkie elementy losowe,
- 25% tablicy posortowane,
- 50% tablicy posortowane,
- 75% tablicy posortowane,
- 95% tablicy posortowane,
- 99% tablicy posortowane,
- $99{,}7\%$ tablicy posortowane,
- $posotowana\ odwrotnie$.

W krótkim menu na początku można wybrać przy pomocy, którego algorytmu chcemy posortować tablice. Znajduje się też tam opcja ponownego wygenerowania tablic. Po wybraniu opcji sortowania rozpoczyna się sortowanie. Czasy są mierzone za pomocą funkcji z biblioteki std::chrono, która pozwala na dokładne pomiary czasu. Zmierzone czasy dla poszczególnych przypadków są zapisywane do pliku o rozszerzeniu „ .csv”(wartości oddzielone przecinkami).

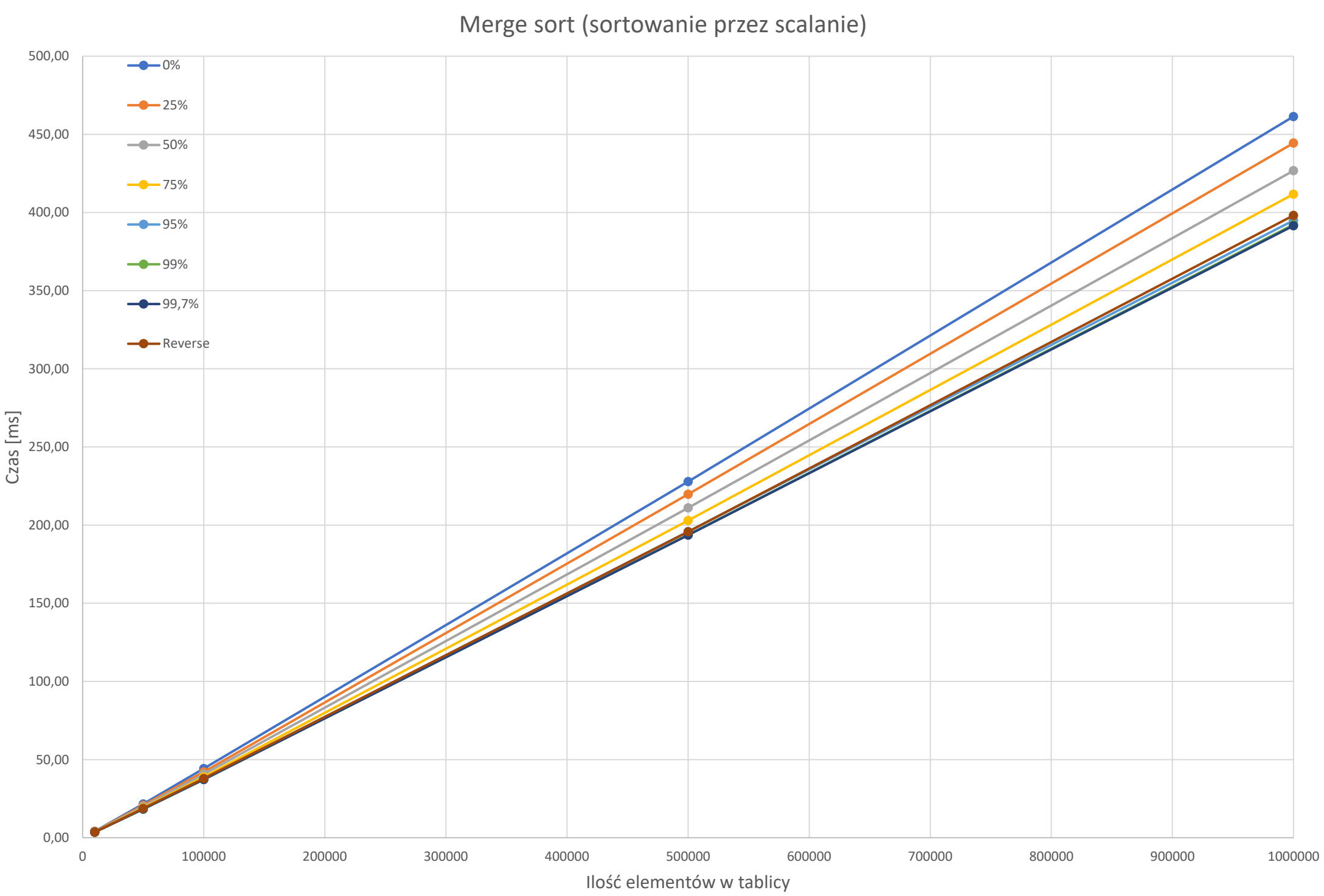
3. Wyniki sortowań

Wyniki czasu sortowania w każdym przypadku można zaobserwować w tabelach poniżej oraz wyniki zostały przedstawione na wykresach.

3.1. Sortowanie przez scalanie

Procent	Ilość danych	Średni czas [ms]	Min czas [ms]	Max czas [ms]
0%	10000	4,11	4,03	4,52
	50000	21,54	21,26	22,41
	100000	44,23	43,46	49,93
	500000	227,85	226,85	232,35
	1000000	461,45	459,84	467,90
25%	10000	3,98	3,93	4,76
	50000	20,77	20,60	23,86
	100000	42,19	42,00	43,61
	500000	219,78	218,89	223,75
	1000000	444,40	442,97	454,34
50%	10000	3,82	3,79	3,93
	50000	19,96	19,85	20,75
	100000	40,62	40,41	42,19
	500000	211,10	210,22	216,64
	1000000	426,76	425,33	429,23
75%	10000	3,68	3,65	3,77
	50000	19,15	19,03	20,29
	100000	38,92	38,69	41,14
	500000	202,92	201,53	239,52
	1000000	411,77	408,31	458,23
95%	10000	3,59	3,55	4,19
	50000	18,52	18,40	19,21
	100000	37,51	37,34	38,24
	500000	195,89	194,47	202,62
	1000000	394,93	393,55	401,60
99%	10000	3,56	3,53	3,65
	50000	18,40	18,28	19,76
	100000	37,31	37,11	38,44
	500000	193,73	192,99	200,38
	1000000	392,28	390,76	403,79
99,70%	10000	3,56	3,52	4,02
	50000	18,38	18,28	19,16
	100000	37,24	37,08	37,82
	500000	193,57	192,78	196,77
	1000000	391,56	390,11	395,90
Reverse	10000	3,57	3,53	4,05
	50000	18,68	18,52	20,64
	100000	37,90	37,69	41,14
	500000	195,70	194,96	201,82
	1000000	398,16	393,14	425,41

Tabela 1. Czasy trwania sortowań w różnych przypadkach dla algorytmu Merge sort.

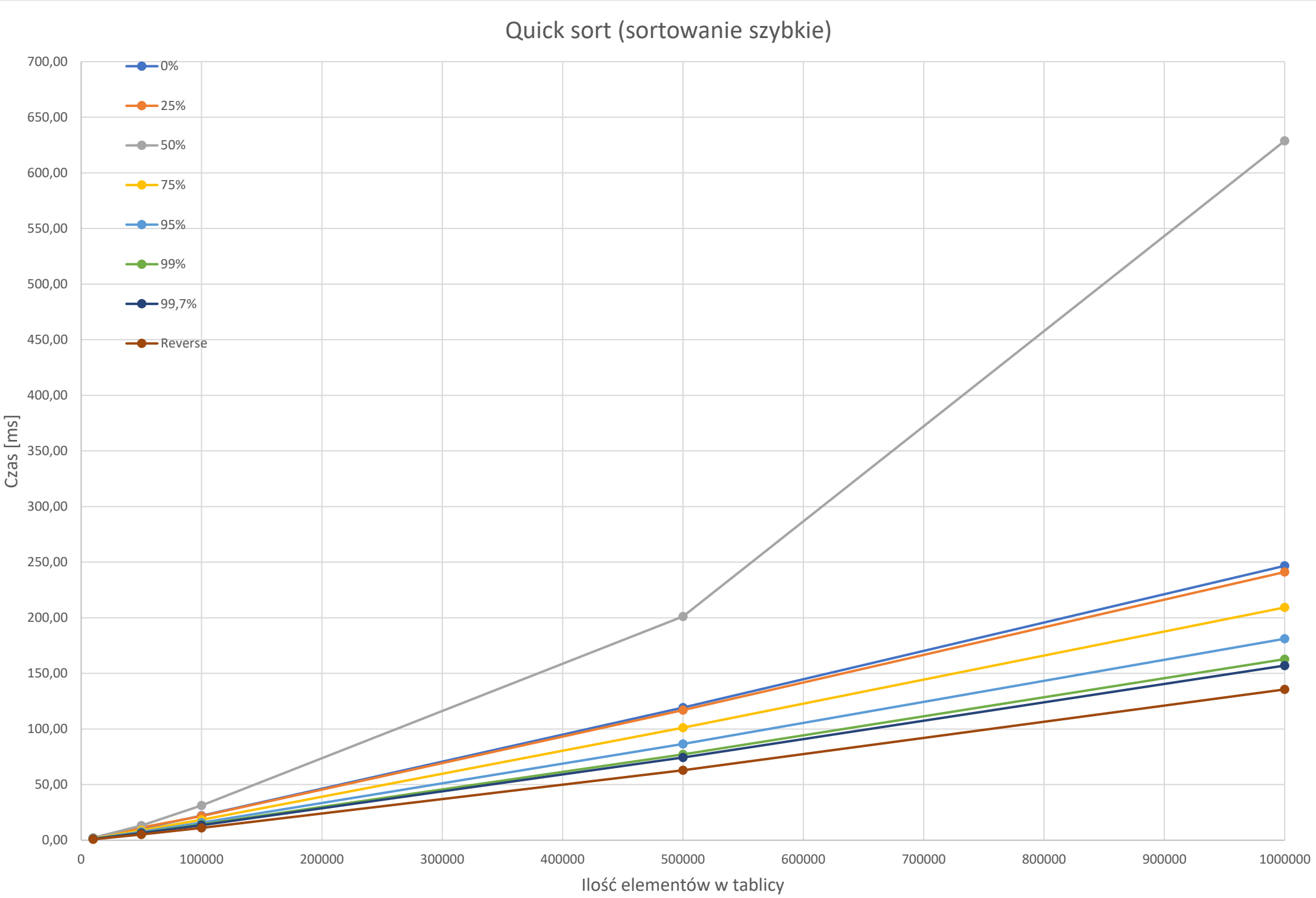


Wykres 1. Prezentacja wyników czasu sortowań dla algorytmu Merge sort.

3.2. Sortowanie szybkie

Procent	Ilość danych	Średni czas [ms]	Min czas [ms]	Max czas [ms]
0%	10000	1,87	1,80	2,34
	50000	10,50	10,21	10,98
	100000	21,98	21,42	23,35
	500000	119,33	116,85	122,64
	1000000	246,71	243,82	269,31
25%	10000	1,84	1,81	1,89
	50000	11,38	10,20	34,86
	100000	21,64	21,32	23,29
	500000	117,02	114,77	119,95
	1000000	241,13	236,42	246,31
50%	10000	2,24	1,72	4,74
	50000	13,11	9,76	32,46
	100000	31,24	20,27	91,14
	500000	201,19	111,42	826,57
	1000000	628,78	230,84	2474,62
75%	10000	1,59	1,52	2,00
	50000	8,92	8,56	11,10
	100000	18,47	17,89	19,56
	500000	101,17	99,38	106,46
	1000000	209,30	206,03	212,76
95%	10000	1,33	1,27	1,63
	50000	7,58	7,31	8,27
	100000	15,84	15,34	19,34
	500000	86,58	85,25	89,62
	1000000	181,16	178,04	199,67
99%	10000	1,17	1,12	1,61
	50000	6,74	6,49	9,06
	100000	14,17	13,72	15,17
	500000	77,20	75,64	82,78
	1000000	162,81	160,78	165,97
99,70%	10000	1,12	1,04	1,67
	50000	6,41	6,19	7,17
	100000	13,60	13,23	14,43
	500000	74,35	72,91	77,10
	1000000	157,05	155,20	160,45
Reverse	10000	0,83	0,81	1,19
	50000	5,11	4,93	6,98
	100000	11,09	10,79	11,89
	500000	63,00	61,81	65,76
	1000000	135,61	133,92	141,77

Tabela 2. Czasy trwania sortowań w różnych przypadkach dla algorytmu Quicksort.

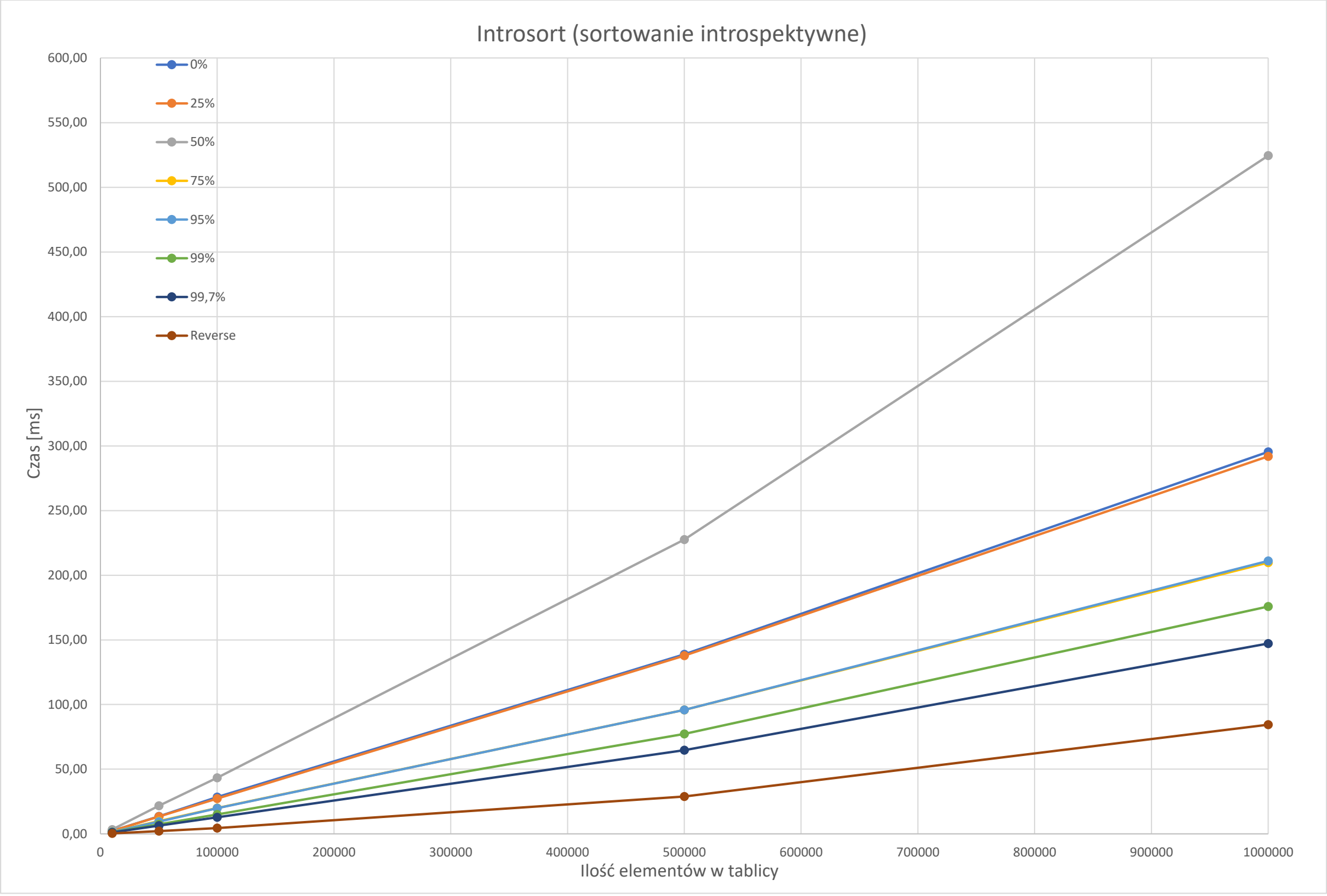


Wykres 2. Prezentacja wyników czasu sortowań dla algorytmu Quicksort.

3.3. Sortowanie introspektywne

Procent	Ilość danych	Średni czas [ms]	Min czas [ms]	Max czas [ms]
0%	10000	2,23	1,94	2,58
	50000	13,51	12,30	15,66
	100000	28,39	24,99	39,34
	500000	138,80	125,17	167,28
	1000000	295,40	264,08	336,21
25%	10000	2,22	1,90	2,74
	50000	13,41	12,07	16,05
	100000	27,27	24,41	32,50
	500000	137,74	120,73	166,53
	1000000	291,95	256,48	349,92
50%	10000	3,30	1,91	5,39
	50000	21,68	11,73	33,80
	100000	43,29	24,65	72,89
	500000	227,52	126,33	409,83
	1000000	524,44	271,83	902,81
75%	10000	1,57	1,42	2,09
	50000	9,91	9,19	10,95
	100000	19,94	18,69	22,24
	500000	95,83	89,34	105,45
	1000000	209,80	197,19	241,12
95%	10000	1,48	1,33	1,94
	50000	9,48	8,52	10,49
	100000	19,74	17,40	21,10
	500000	95,80	88,51	109,43
	1000000	211,08	197,64	246,65
99%	10000	1,26	1,16	1,40
	50000	7,38	6,61	8,08
	100000	15,02	13,49	16,46
	500000	77,33	73,96	93,33
	1000000	175,81	168,40	186,21
99,70%	10000	1,11	0,92	1,54
	50000	6,37	5,89	7,02
	100000	12,85	12,26	13,92
	500000	64,65	61,79	86,46
	1000000	147,20	142,33	152,79
Reverse	10000	0,34	0,34	0,39
	50000	2,19	2,14	3,18
	100000	4,48	4,35	5,13
	500000	28,89	28,30	31,62
	1000000	84,42	82,93	99,09

Tabela 3. Czasy trwania sortowań w różnych przypadkach dla algorytmu Introsort.



Wykres 3. Prezentacja wyników czasu sortowań dla algorytmu Introsort.

4. Wnioski

Sortowanie introspektywne i szybkie wyglądają podobnie, bo opierają się na taj samej zasadzie działania.

Można jednak zauważyć, że dla 50% posortowanej tablicy czas sortowania mocno odstaje od pozostałych. Jest to związane z doborem pivot'a w funkcji Partition. Bowiem jest on zawsze środkowym elementem tablicy, zatem jest to element o największej wartości w jednej z podtablic. Zatem występuje najgorszy przypadek dla sortowania szybkiego. Ten pesymistyczny przypadek nie ma aż tak drastycznego rezultatu w przypadku sortowania introspektywnego.

Ku zaskoczeniu introsort nie jest w każdym przypadku szybszy od quicksorta, zgodnie z założeniami. Co prawda dla odwrotnie posortowanej i dla w połowie posortowanej jest szybszy, czyli spełnia założenie o wyeliminowaniu najgorszego przypadku. Dla reszty przypadków. Jest wolniejszy lub zbliżony. Wartości głębokości rekurencji jest ustalona na $2 \cdot \log n$, a wartość progu, przy którym algorytm zmienia się na sortowanie przez wstawianie to 16. Zmienianie wartości prowadziło do przedłużenia czasu sortowania. Dłuższy czas może być spowodowany złym doborem wartości parametrów.

Sortowanie przez scalanie zgodnie z przewidywaniami nie prezentuje dużych różnic podczas eksperymentów. Spowodowane jest to naturą tego algorytmu, gdyż dane wejściowe nie mają na niego wpływu. Na pozostałe omówione algorytmy ma.

5. Bibliografia

- https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
- https://pl.wikipedia.org/wiki/Sortowanie_szybkie
- http://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html
- https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
- https://www.geeksforgeeks.org/merge-sort/
- https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
- https://pl.wikipedia.org/wiki/Sortowanie_przez_wstawianie

6. Sprzęt

CPU – Intel Corei7-8750H 2.20GHz
GPU – GTX 1050
RAM – 16GB
SYSTEM – Windows 10 Home 64-bitowy