Politechnika Wrocławska

Projektowanie Algorytmów i Metod Sztucznej Inteligencji

Imię i nazwisko prowadzącego	dr inż. Łukasz Jeleń
Termin zajęć	środa, $11^{15} - 13^{00}$
Data oddania sprawozdania	3.04.2019

Temat projektu

Algorytmy sortujące

Imię i Nazwisko	**Indeks**
Anna Bielecka	241159

1. Wstęp

Tematem projektu było zaimplementowanie algorytmów sortujących: sortowanie szybkie, sortowanie przez scalanie oraz sortowanie introspektywne. Test efektywności algorytmów należało przeprowadzić mierząc czas sortowania tablic o 10000, 50000, 100000, 500000, 1000000 elementów, o różnym stopniu posortowania 0%, 25%, 50%, 75%, 95%, 99%, 99.7% oraz tablicy posortowanej odwrotnie.

Do pomiaru czasu w milisekundach (we wszystkich pomiarach) została użyta biblioteka <ctime> oraz następująca funkcja:

```
StartTime = clock();
Funkcja, której czas wykonania jest mierzony
StopTime = clock();
czas= ((double)(StopTime - StartTime) /CLOCKS_PER_SEC)*1000;
```

2. Sortowanie szybkie (quicksort)

Sortowanie szybkie jest algorytmem rekurencyjnym. W podanej metodzie oryginalna tablica dzielona jest na dwie podtablice, z których pierwsza zawiera elementy mniejsze lub równe od pewnego wybranego klucza, nazywanego elementem osiowym. Druga natomiast zawiera elementy większe lub równe od wspomnianego wyżej elementu. Następnie otrzymane podtablice poddaje się podobnemu procesowi podziału. Proces dzielenia powtarzany jest do momentu, gdy uzyskamy wyłącznie tablice jednoelementowe, których nie trzeba już sortować. Element osiowy nie bierze udziału w sortowaniu, ponieważ już jest na swojej pozycji.

Złożoność obliczeniowa zależy od wyboru elementu osiowego.

Przypadek optymistyczny zachodzi gdy element osiowy dzieli tablicę na dwie podtablice, których wielkość w przybliżeniu wynosi $\frac{n}{2}$, co prowadzi do $log_2 n$ wywołań rekurencyjnych, każde z nich operuje na maksymalnym rozmiarze tablicy (n), co prowadzi do złożoności $O(n * logn)$.

Przypadek średni złożoność obliczeniowa jest zbliżona do przypadku optymistycznego.

Przypadek pesymistyczny zachodzi gdy przy każdym wywołaniu funkcji jako element osiowy wybierany jest najmniejszy (lub największy) element tablicy. Złożoność obliczeniowa w przypadku pesymistycznym wynosi $O(n^2)$.

3. Sortowanie przez scalanie (mergesort)

Sortowanie przez scalanie - podobnie jak sortowanie szybkie - to metoda rekurencyjna z gatunku "dziel i zwyciężaj". Jednak w odróżnieniu do algorytmu sortowania szybkiego, mergesort w każdym przypadku osiąga złożoność obliczeniową $O(n * logn)$. Ideą działania algorytmu jest dzielenie zbioru danych na mniejsze zbiory,

aż do uzyskania (n) zbiorów jednoelementowych, gdzie każdy z tych podzbiorów będzie posortowany. Następnie zbiory te są łączone w coraz większe posortowane zbiory, aż do uzyskania jednego, posortowanego zbioru n -elementowego.

Dużą wadą algorytmu jest zużycie pamięci z powodu konieczności zastosowania dodatkowej tablicy pomocniczej.

W przypadku tablic liczb całkowitych nie jest to bardzo istotne, ale przy sortowaniu bardziej złożonych danych ten koszt może być znaczący i wpływać negatywnie na efektywność sortowania.

4. Sortowanie introspektywne

Sortowanie introspektywne jest sortowaniem hybrydowym tzn. zawiera w sobie więcej niż jeden algorytm sortowania. Bazuje ono na sortowaniu szybkim i sortowaniu przez kopcowanie, pozwalając przy tym wyeliminować złożoność obliczeniową $O(n^2)$ dla pesymistycznego przypadku sortowania szybkiego. W procedurze głównej ustala się zmienną maksymalną określającą dozwoloną głębokość wywołań rekurencyjnych (w tym przypadku będzie to $2 * \log_2 n$).

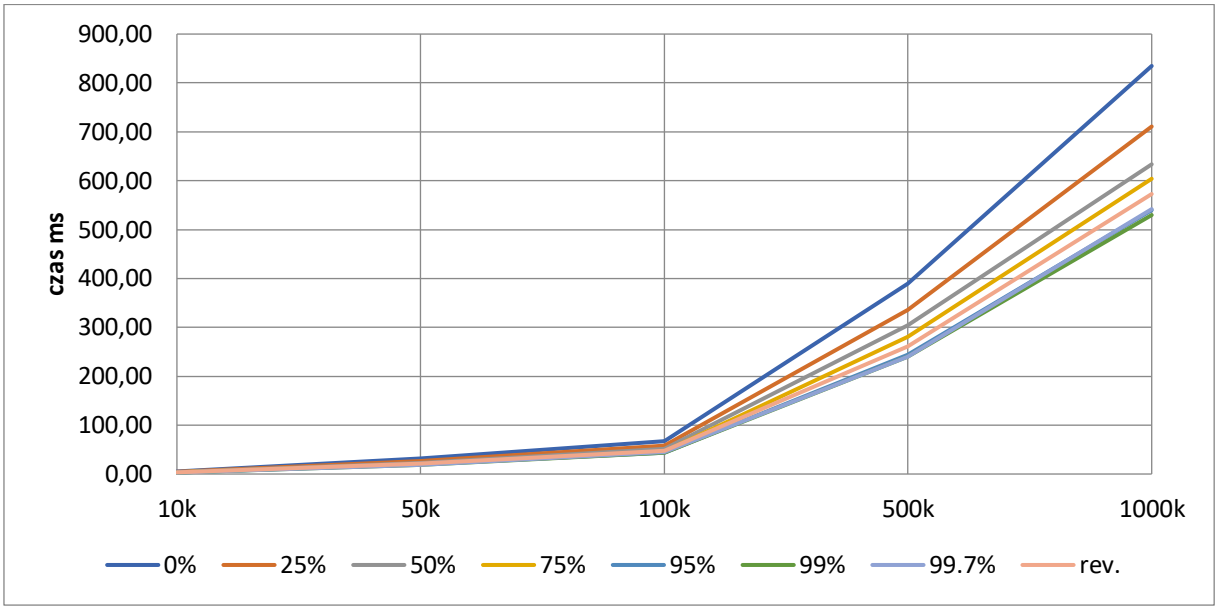
Początkowo sortowanie introspektywne działa jak sortowanie szybkie, lecz każde rekurencyjne wywołanie powoduje zmniejszenie zmiennej określającą maksymalną dozwoloną głębokość wywołań rekurencyjnych o 1. Gdy zmienna ta osiągnie wartość 0 wywołania rekurencyjne są kończone i dla aktualnie wykorzystywanej podtablicy wywoływane jest sortowanie przez kopcowanie.

Taki algorytm pozwala na wykorzystanie najlepszych cech obu sortowań. Sortowanie szybkie dzieli tablice na małe podzbiory, które następnie są kopcowane. Złożoność obliczeniowa kopcowania to $O(n * logn)$., jednak dla dużych podzbiorów jest ono kilkukrotnie wolniejsze niż sortowanie szybkie. Sortowanie introspektywne wywoływane dla małych tablic, pozwala na otrzymanie podobnych czasów sortowania, co w średnim przypadku sortowania szybkiego, z jednoczesną eliminacja jego najgorszego przypadku.

5. Wyniki pomiarów

		10k	50k	100k	500k	1000k
0,00%	min	5,00	30,00	65,00	380,00	794,00
	max	7,00	36,00	73,00	456,00	981,00
	śr.	5,24	31,16	66,72	389,28	834,76
25,00%	min	4,00	25,00	56,00	330,00	694,00
	max	7,00	29,00	68,00	349,00	817,00
	śr.	4,62	26,55	58,35	335,78	710,86
50,00%	min	3,00	22,00	48,00	290,00	615,00
	max	6,00	28,00	66,00	431,00	741,00
	śr.	3,86	22,90	51,64	304,13	633,55
75,00%	min	3,00	19,00	43,00	270,00	587,00
	max	5,00	22,00	59,00	351,00	722,00
	śr.	3,41	20,31	46,29	280,46	604,02
95,00%	min	2,00	18,00	41,00	239,00	525,00
	max	4,00	29,00	55,00	300,00	646,00
	śr.	3,00	19,54	43,95	244,46	539,76
99,00%	min	2,00	18,00	41,00	235,00	516,00
	max	4,00	20,00	71,00	294,00	594,00
	śr.	2,97	18,60	43,35	240,07	529,79
99,70%	min	2,00	18,00	41,00	235,00	514,00
	max	5,00	25,00	82,00	290,00	842,00
	śr.	2,96	18,61	45,30	240,60	541,87
rev.	min	3,00	20,00	45,00	257,00	556,00
	max	5,00	28,00	66,00	305,00	693,00
	śr.	3,40	21,17	47,33	261,20	572,78

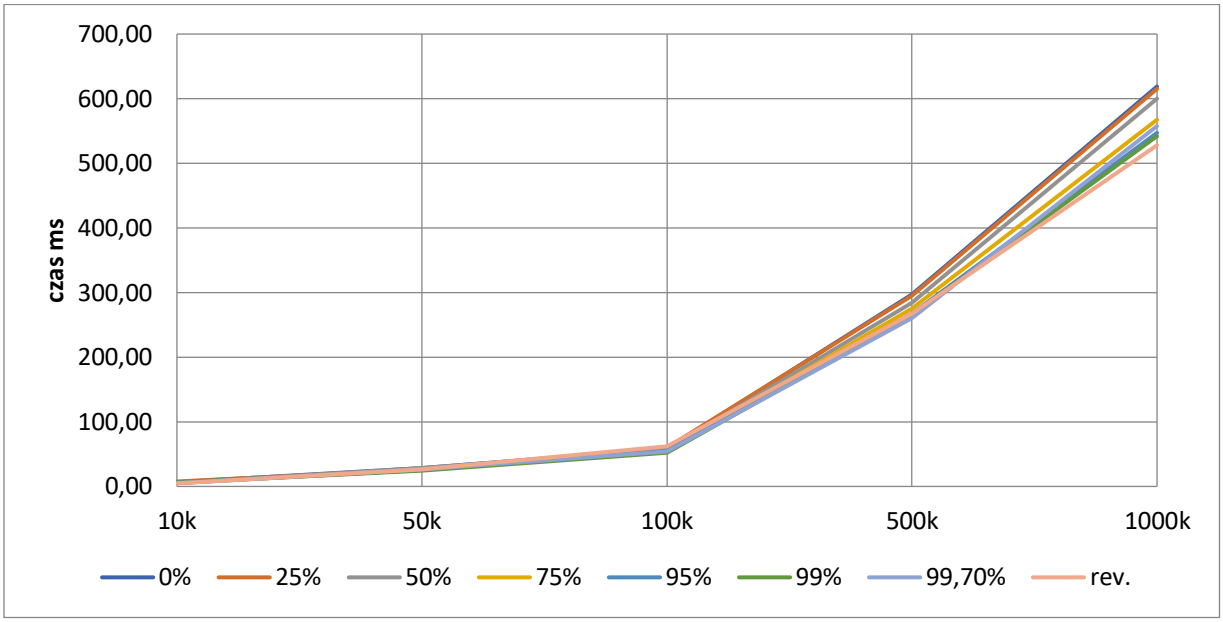
Tabela 1: Wyniki minimalne, maksymalne i wartość średnia pomiaru czasu dla sortowania szybkiego.



Rysunek 1: Wykres zależności czasu od ilości danych dla sortowania szybkiego.

		10k	50k	100k	500k	1000k
0,00%	min	5,00	26,00	52,00	283,00	583,00
	max	10,00	42,00	97,00	436,00	784,00
	śr.	6,58	27,92	57,99	296,83	618,90
25,00%	min	5,00	25,00	51,00	274,00	558,00
	max	14,00	30,00	111,00	412,00	922,00
	śr.	7,58	26,57	59,99	294,46	615,34
50,00%	min	4,00	24,00	49,00	264,00	540,00
	max	10,00	45,00	97,00	435,00	1779,00
	śr.	5,10	28,20	57,47	284,36	600,12
75,00%	min	4,00	23,00	47,00	256,00	521,00
	max	9,00	30,00	123,00	440,00	997,00
	śr.	5,17	25,33	54,69	274,62	567,49
95,00%	min	4,00	22,00	46,00	249,00	501,00
	max	10,00	47,00	112,00	419,00	859,00
	śr.	6,14	25,49	55,10	267,17	547,14
99,00%	min	4,00	22,00	46,00	249,00	502,00
	max	10,00	26,00	120,00	437,00	800,00
	śr.	5,86	24,37	52,41	264,98	542,06
99,70%	min	4,00	22,00	46,00	247,00	501,00
	max	8,00	47,00	141,00	307,00	1008,00
	śr.	4,94	26,11	54,50	259,88	557,65
rev.	min	4,00	23,00	47,00	245,00	503,00
	max	8,00	46,00	145,00	456,00	748,00
	śr.	4,76	26,55	61,69	266,79	527,89

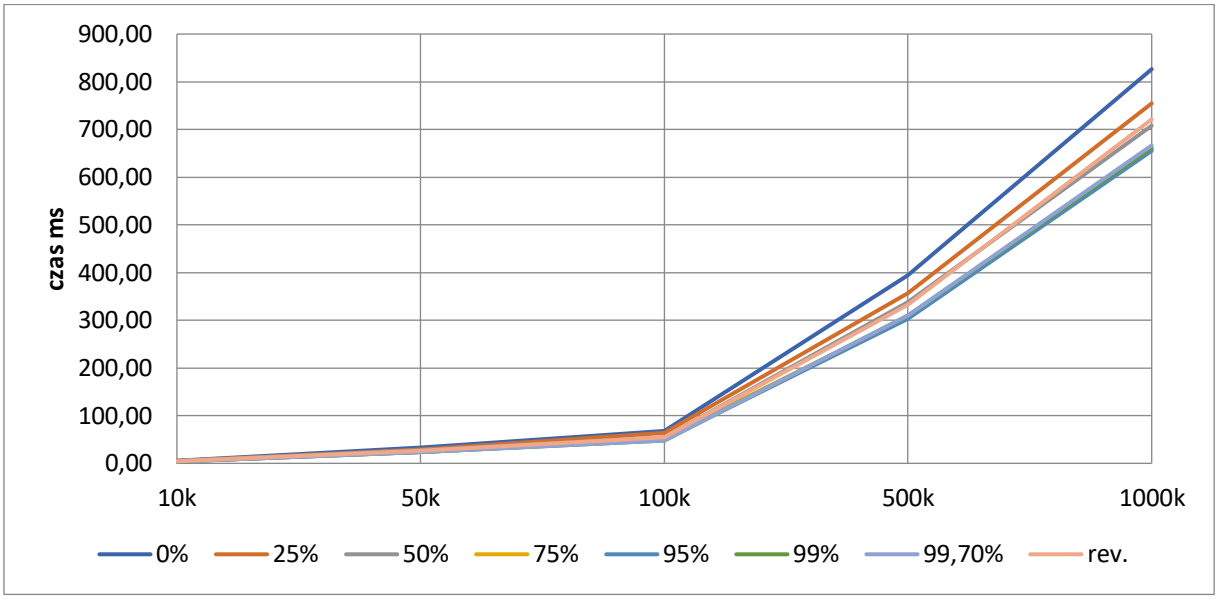
Tabela 2: Wyniki minimalne, maksymalne i wartość średnia pomiaru czasu dla sortowania przez scalanie.



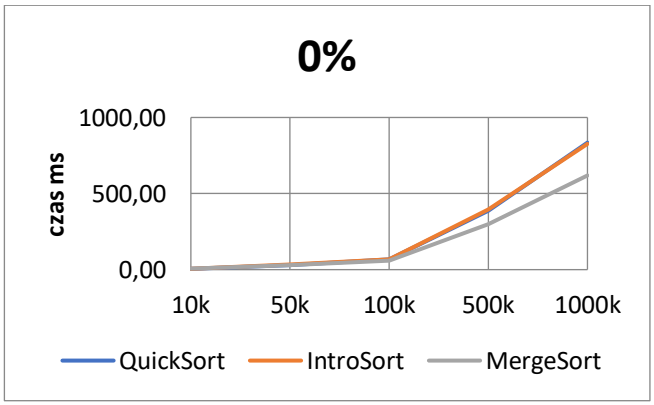
Rysunek 2: Wykres zależności czasu od ilości danych dla sortowania przez scalanie.

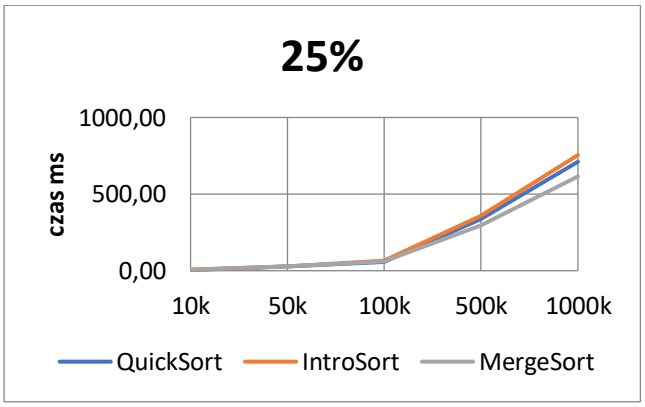
		10k	50k	100k	500k	1000k
0,00%	min	5,00	32,00	65,00	378,00	803,00
	max	8,00	45,00	103,00	511,00	997,00
	śr.	5,58	33,48	68,01	394,25	826,37
25,00%	min	4,00	27,00	58,00	343,00	726,00
	max	7,00	35,00	96,00	496,00	997,00
	śr.	4,94	28,91	63,86	356,09	754,71
50,00%	min	4,00	25,00	52,00	321,00	683,00
	max	9,00	41,00	82,00	442,00	839,00
	śr.	5,17	27,61	54,11	338,26	707,96
75,00%	min	3,00	24,00	52,00	296,00	636,00
	max	5,00	35,00	81,00	429,00	782,00
	śr.	4,01	25,45	53,15	306,34	658,62
95,00%	min	3,00	22,00	47,00	296,00	634,00
	max	6,00	38,00	75,00	432,00	889,00
	śr.	3,91	23,89	48,85	303,29	655,13
99,00%	min	3,00	22,00	47,00	300,00	642,00
	max	4,00	32,00	71,00	449,00	836,00
	śr.	3,90	23,11	47,95	309,29	661,38
99,70%	min	3,00	22,00	46,00	300,00	644,00
	max	5,00	37,00	73,00	417,00	871,00
	śr.	3,95	23,82	47,93	310,16	666,64
rev.	min	4,00	24,00	51,00	321,00	685,00
	max	7,00	37,00	88,00	454,00	978,00
	śr.	4,48	25,68	54,02	332,66	721,19

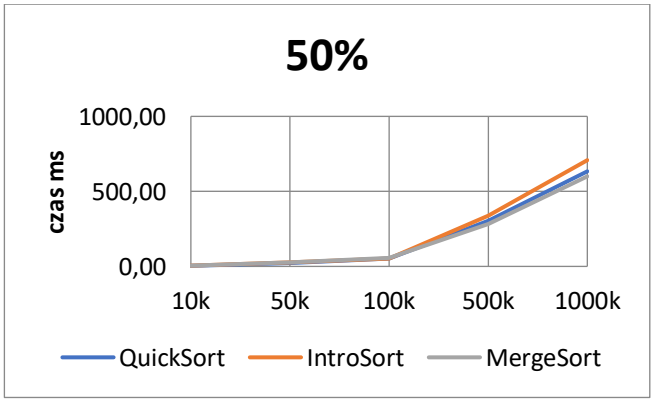
Tabela 3: Wyniki minimalne, maksymalne i wartość średnia pomiaru czasu dla sortowania szybkiego.

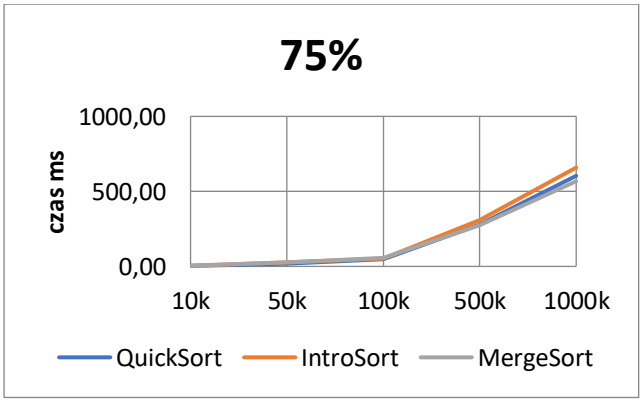


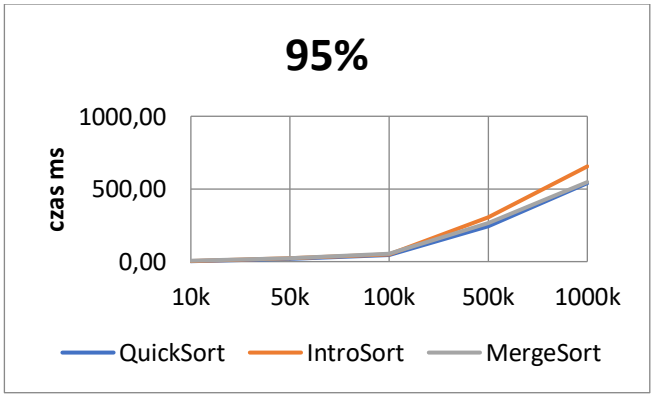
Rysunek 3: Wykres zależności czasu od ilości danych dla sortowania introspektywnego.

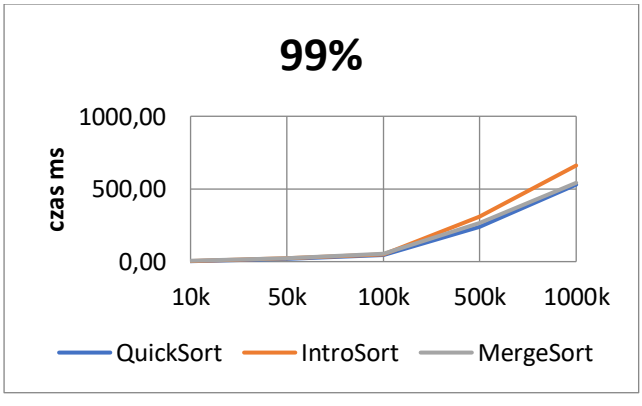


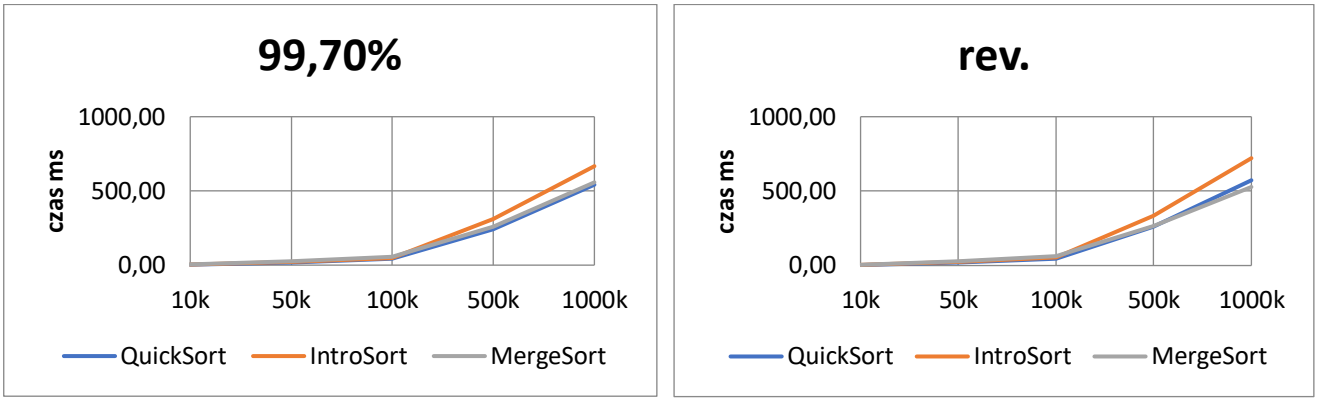












Rysunek 4: Porównanie wykresów zależności czasu od ilości danych dla poszczególnych algorytmów sortowania.

6. Wnioski

Z uzyskanych wyników można zauważyć, że wszystkie algorytmy sortowania działają bardzo szybko.

Ze zbiorami o dużej liczebności najlepiej radziło sobie sortowanie przez scalanie, natomiast najdłuższy czas sortowania osiągnęło sortowanie introspektywne (chociaż w przypadku zbioru z zerowym wstępnym posortowaniem elementów jest porównywalne z sortowaniem szybkim).

Wstępne posortowanie początkowych elementów tabel ma największy wpływ przy sortowaniu szybkim (im większy procent, tym lepsze wyniki sortowania. Wynika to ze sposobu wyboru elementu osiowego, w przypadku zaimplementowanego algorytmu wspomniany element wybierany był jako element środkowy podprzedziału (jeżeli podprzedział zawierał wartości posortowane to element osiowy dzieli go na dwa inne podprzedziały o zbliżonej długości co jest efektem pożądanym). Im większy procent zbioru został wstępnie posortowany tym bardziej czasy uzyskane dla sortowania szybkiego zbliżały się do wartości uzyskanych przy sortowaniu przez scalanie (które okazało się działać najszybciej).

Literatura

1. Drozdek A., *C++. Algorytmy i struktury danych*, Helion
2. Wróblewski P., *Algorytmy, struktury danych i techniki programowania*, Wydanie VI, Helion
3. Cormen T., Leiserson C.E., Rivest R.L., Stein C., *Wprowadzenie do algorytmów*, WNT
4. http://informatyka.wroc.pl/node/433?page=0,1
5. https://pl.wikipedia.org/wiki/Sortowanie_szybkie
6. https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
7. https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
8. https://pl.wikipedia.org/wiki/Sortowanie_introspektywne