Projektowanie algorytmów i metody sztucznej inteligencji, bud C-3,s.103

Temat projektu: Algorytmy sortowania

Projekt wykonał:

Jakub Kędzierski 248915

Data: 26.03.2020r

Termin zajęć: czwartek 9:15-11:00

Prowadzący: Dr inż. Łukasz Jeleń

1. Wstęp

W projekcie przedstawiono 3 metody sortowania:

- sortowanie szybkie - quicksort
- sortowanie przez scalanie - merge sort
- sortowanie introspektywne - introsort (ten algorytm do pracy potrzebował dwóch kolejnych, nie uwzględnionych w opisie).

Kod źródłowy algorytmów sortowania został napisany w języku C++ i skonstruowany tak, aby było możliwe sortowanie różnego typu danych (użyto szablonów). Dla każdego algorytmu przeprowadzono testy, których celem było eksperymentalne określenie złożoności obliczeniowej, i czasowej. Następnie porównano wyniki przeprowadzonych badań z założeniami teoretycznymi.

Testy przeprowadzono dla 100 tablic z elementami typu całkowitoliczbowego o rozmiarach 10 000, 50 000, 100 000, 500 000, 1 000 000. Dla każdego algorytmu przetestowano różne warianty wstępnego ułożenia tablicy dla określenia najlepszego, najgorszego oraz średniego przypadku złożoności obliczeniowej algorytmu. Przetestowane warianty to tablica wstępnie uporządkowana w: 0%, 25%, 50%, 75%, 95%, 99%, 99.7% oraz uporządkowana w odwrotnej kolejności.

2. Opis algorytmów

2.1. Sortowanie szybkie

Algorytm działa na zasadzie „dziel i zwyciężaj”: podział problemu na mniejsze problemy, rozwiązanie tych problemów i połączenie wyników. Przebieg algorytmu sortowania przy sortowaniu tablicy:

- wybranie elementu który rozdziela nasze dane (piwot),

- dzielenie danych na większe i mniejsze lub równe piwotu,

- ponowne sortowanie wcześniej podzielonych danych, aż do momentu gdy do posortowania zostaje jeden element.

Już z przebiegu algorytmu widać że opiera się on na ciągłej rekurencji, co w przypadku niekorzystnego wyboru piwotu i niefortunnego ułożenia danych (podczas testów ten przypadek pojawiał się dla wstępnie posortowanej tablicy w 50%) prowadzi do znacznie większej złożoności obliczeniowej niż standardowo się zakłada. Algorytm sprawdza się dla danych ułożonych zgodnie z równomiernym rozkładem prawdopodobieństwa i dla średniej wielkości tablic (z testów maksymalnie 100-500 tyś elementów).

Złożoność obliczeniowa to dla przypadku:

-średniego : $O(n \cdot \log(n))$,

-najgorszego : $O(n^2)$.

2.2. Sortowanie przez scalanie

Podobnie jak algorytm sortowania szybkiego, sortowanie przez scalanie opiera się na zasadzie „dziel i zwyciężaj”. Przebieg algorytmu:

- podział danych na dwie równe części,

- jeśli podzielona część zawiera więcej niż jeden element ponownie dzielimy dane,

- gdy podzielone części zawierają po jednym elemencie scalamy dane zgodnie z naszą chęcią posortowania (rosnąco lub malejąco).

Można wnioskować że część przebiegu algorytmu jest podobna do sortowania szybkiego. Również dokonujemy podziału, który w przypadku sortowania przez scalanie jest równomierny, niezależny od piwotu. Analogicznie sortujemy dopóki nie otrzymamy jednoelementowych ciągów. Różnicą jest podstawa algorytmu czyli scalanie tablic, które wymaga od nas przepisywania danych do tablic pomocniczych. Zaletą algorytmu jest mały wpływ początkowego ułożenia danych na przebieg sortowania, złożoność obliczeniowa dla przypadku pesymistycznego i średniego należy do tej samej klasy. Sortowanie przez scalanie sprawdza się również zarówno dla małej jak ogromnej ilości danych (w testach zarówno 10 000 jak 1 000 000 elementów).

Złożoność obliczeniowa to dla przypadku:

-średniego : $O(n \cdot \log(n))$,

-najgorszego : $O(n \cdot \log(n))$.

2.3. Sortowanie introspektywne

Sortowanie introspektywne to sortowanie hybrydowe, które w swoim działaniu korzysta z 3 innych algorytmów: sortowania szybkiego, sortowania przez kopcowanie, sortowania przez wstawianie. Przebieg algorytmu można przestawić jako przełączanie się na jeden z 3 trybików:

- zaczynamy sortowanie od sortowania szybkiego, a tak naprawdę od znajdowania piwotu i dzielenia na partycje(część nie większą i większą od piwotu),

- gdy liczba elementów jest mniejsza niż 16 przełączamy się na tryb sortowania przez wstawianie, które pomimo dużej złożoności obliczeniowej ($O(n^2)$) dla małej liczby elementów do posortowania jest wysoce efektywny,

- gdy sortowanie wskoczy w zbyt duża rekurencję przełączamy się na sortowanie przez kopcowanie (moment przełączenia oblicza się na podstawie logarytmu liczby elementów pozostałych do posortowania).

Krótki opis sortowania przez wstawianie i przez kopcowanie:

- Algorytm sortowania przez wstawianie polega na wyciąganiu kolejno liczb z ciągu nieposortowanego i wklejaniu w odpowiedni miejsce ciągu posortowanego. W tym algorytmie występuje ciągłe porównywanie (przy wklejaniu liczb), jednak dla małej ilości danych jest to wydajny sposób sortowania.
- Algorytm sortowania przez kopcowanie składa się z dwóch etapów: tworzenie kopca binarnego i sortowanie. Po utworzeniu kopca element początkowy jest zamieniany z ostatnim elementem, elementy posortowane nie są już częścią kopca. Te czynności powtarzamy do posortowania wszystkich elementów.

Sortowanie introspektywne pozwala na uniknięcie najgorszego przypadku występującego podczas sortowania szybkiego gdy złożoność obliczeniowa sięgała $O(n^2)$. Najlepiej sprawdza się dla zestawu danych zgodnych z równomiernym rozkładem prawdopodobieństwa i dla średniej wielkości tablic (do max 500tyś elementów).

Złożoność obliczeniowa to dla przypadku:

-średniego : $O(n \cdot \ln(n))$

-najgorszego : $O(n \cdot \ln(n))$.

3. Wyniki eksperymentów.

Na podstawie danych z sortowania 100 tablic wyliczono średni, najlepszy oraz najgorszy (śr, min, max) czas wykonywania operacji. W tabeli czas sortowania podawano w ms.

Sortowanie szybkie

Liczba elem. (w tyś)	10			50			100			500			1000		
↓ Procent posortowania	śr	min	max	śr	min	max	śr	min	max	śr	min	max	śr	min	max
0	2,0	1,5	5,3	10,9	8,7	17	20	19	26	123	117	137	327	304	363
25	1,8	1,5	4,9	10	8,5	15	19	18	22	123	115	149	328	290	532
50	4,3	1,6	19	38	9,2	178	82	19	372	771	115	5847	1665	292	12144
75	2,1	1,4	9,6	9,2	7,3	13	16	15	18	104	99	114	271	259	294
95	1,9	1,5	4,2	8,1	7,4	11	17	16	20	106	100	116	275	265	321
99	1,5	1,2	2,8	6,9	6,4	9	14	13	16	94	91	102	251	244	270
99,7	1,4	1,1	3,4	6,9	6,0	13	13	12	26	89	86	97	242	235	258
w odwrotnej kolejności	1,1	1,0	2,0	7,4	5,5	32	11	11	21	84	77	99	221	213	251

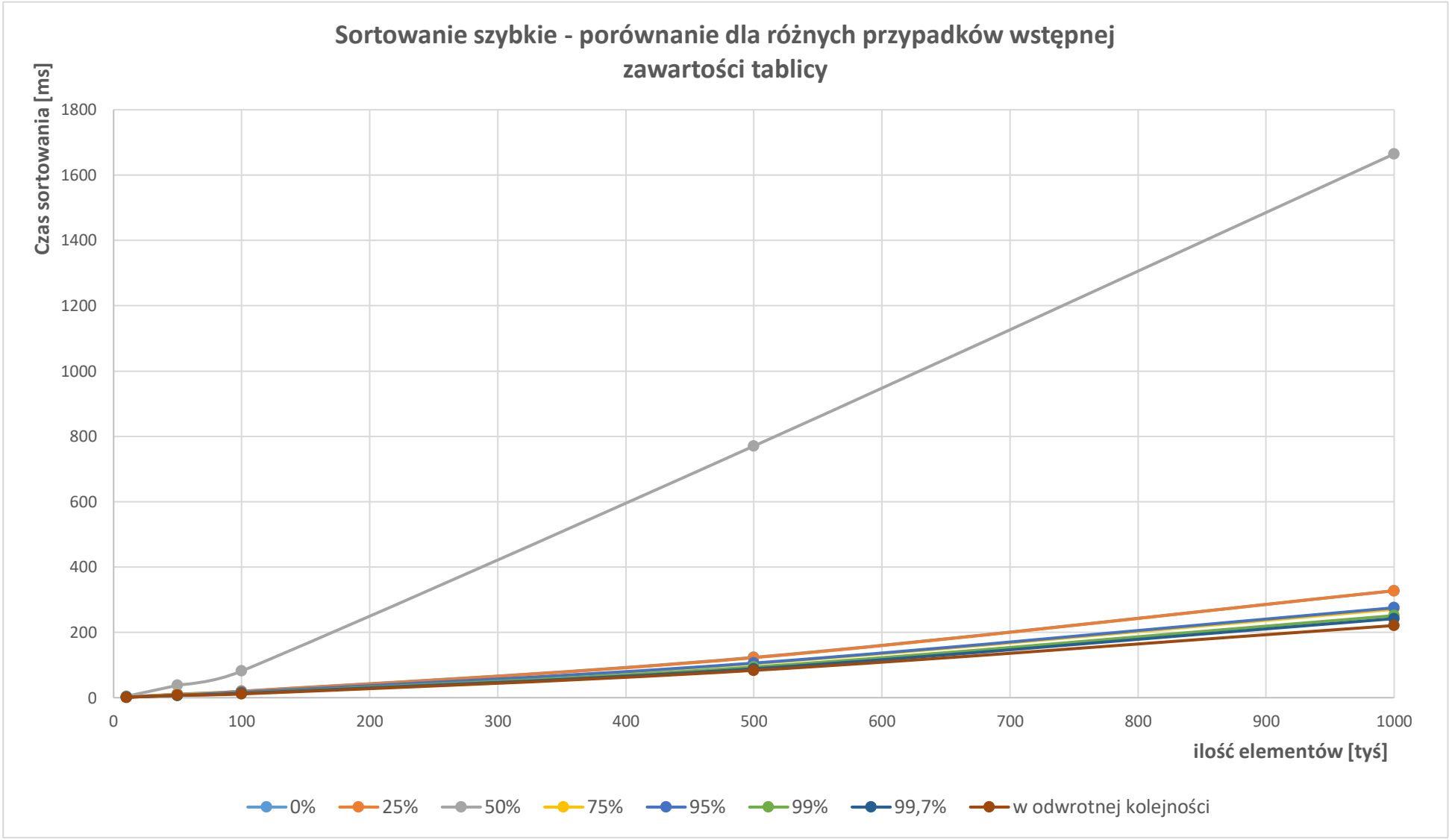
Sortowanie przez scalanie

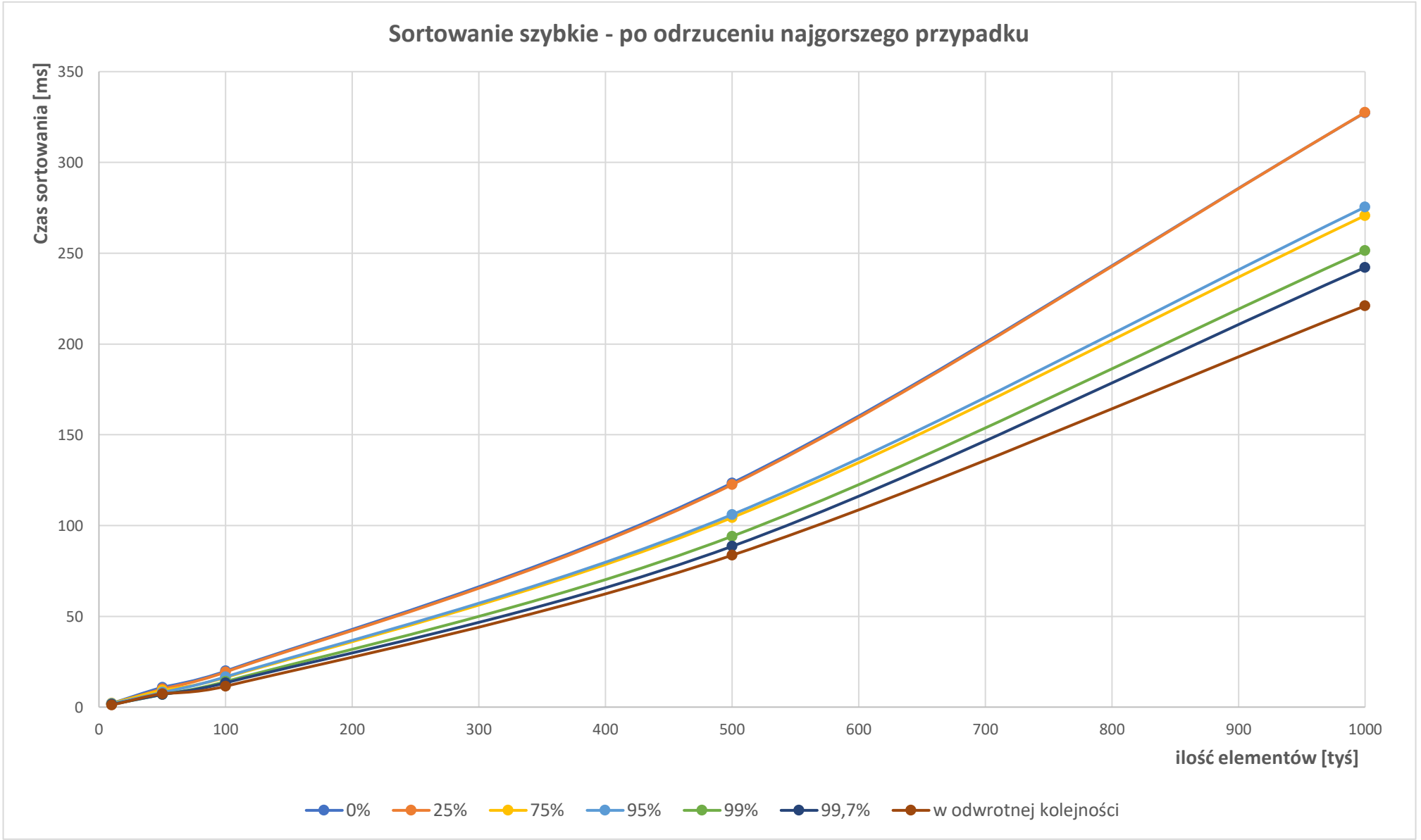
Liczba elem. (w tyś)	10			50			100			500			1000		
↓ Procent posortowania	śr	min	max	śr	min	max	śr	min	max	śr	min	max	śr	min	max
0	2,0	1,6	5,6	11,4	9,4	20	23	20	36	120	113	172	266	241	908
25	1,7	1,4	3,3	12,2	8,6	27	21	18	83	108	102	129	231	212	313
50	1,5	1,2	2,6	8,4	7,3	10	17	16	20	96	90	113	197	181	228
75	1,2	1,0	1,9	7,1	6,1	10	14	13	16	84	77	96	161	145	230
95	1,1	0,9	2,0	6,1	5,3	9	12	11	14	73	66	105	177	132	404
99	1,0	0,9	1,8	7,0	5,2	12	12	11	15	68	65	76	158	129	307
99,7	1,0	0,9	1,7	6,1	5,1	9	12	11	13	68	65	85	148	124	247
w odwrotnej kol.	1,0	0,9	1,8	6,3	5,4	9	13	11	15	68	65	81	150	124	570

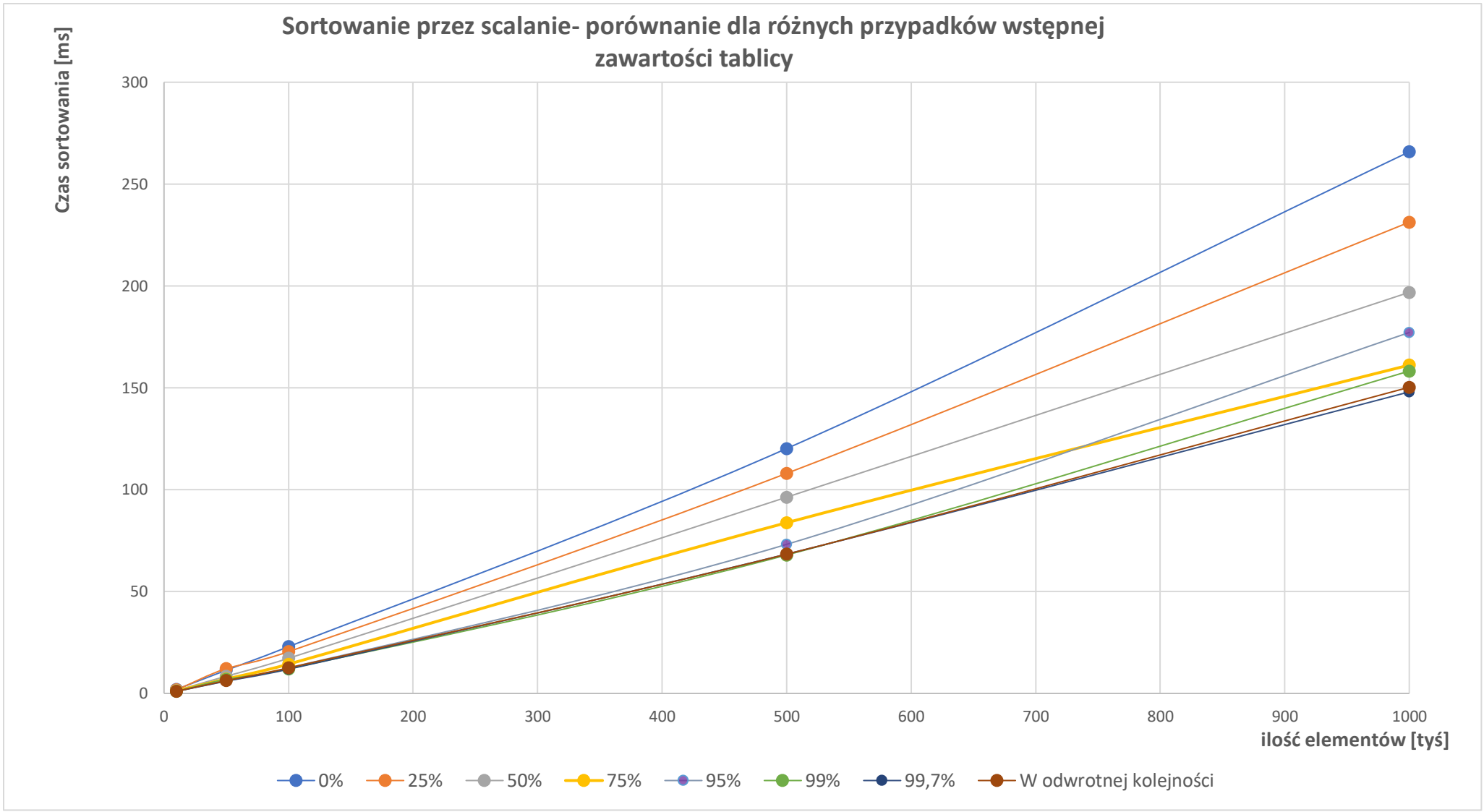
Sortowanie introspektywne

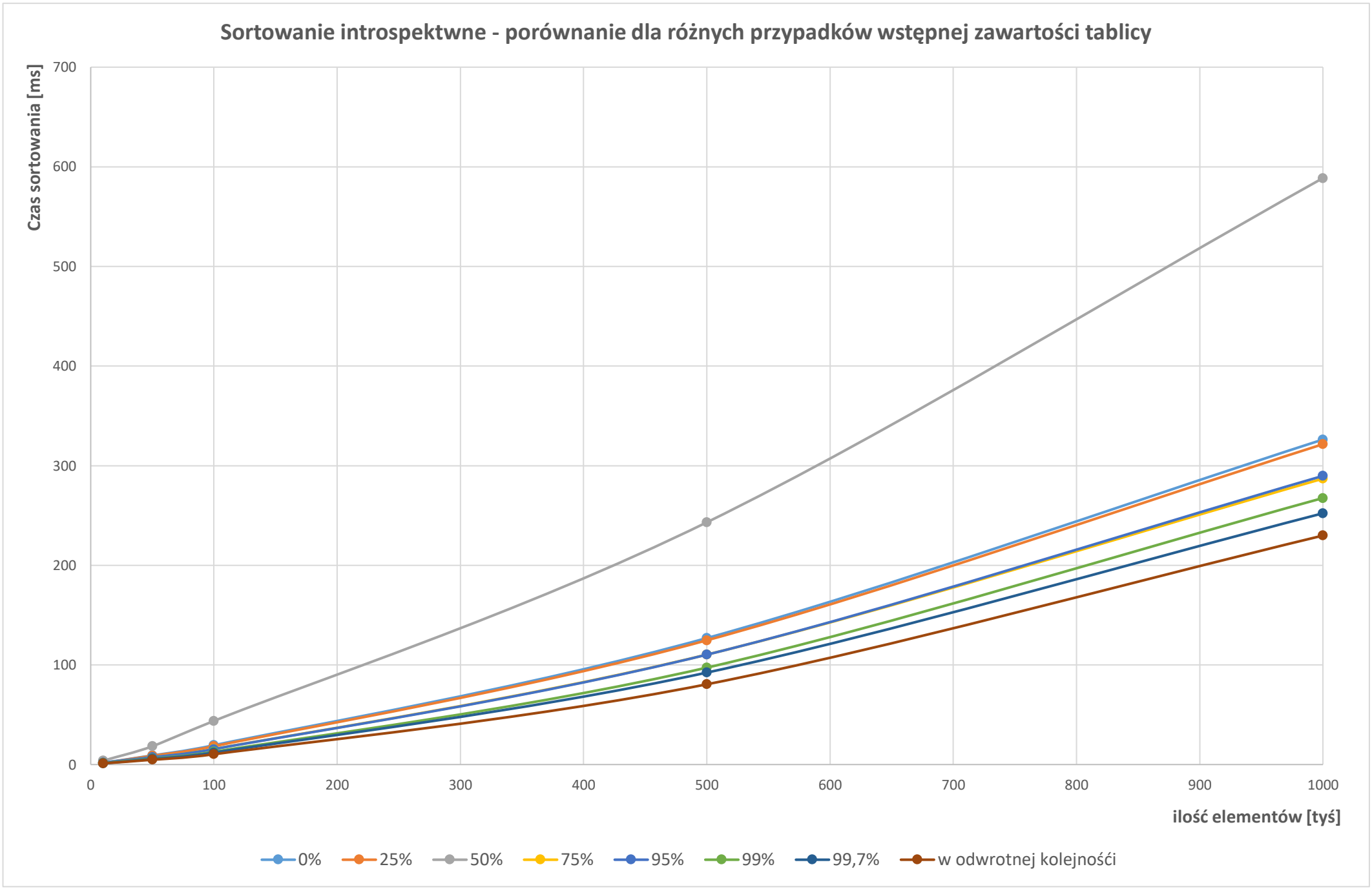
Liczba elem. (w tyś)	10			50			100			500			1000		
↓ Procent posortowania	śr	min	max	śr	min	max	śr	min	max	śr	min	max	śr	min	max
0	1,7	1,4	5,4	9,1	8,4	12	19	17	28	127	120	149	326	306	386
25	1,5	1,4	2,1	8,8	8,0	11	18	17	21	125	115	140	322	309	344
50	3,8	1,5	11,8	18,3	8,1	38	44	18	80	243	122	487	589	307	1058
75	1,6	1,3	2,4	7,5	6,7	9	16	14	18	110	104	120	287	274	307
95	2,0	1,2	10,0	7,4	6,9	10	15	14	18	110	106	124	290	276	332
99	1,5	1,0	5,6	6,0	5,8	7	13	12	14	97	93	107	268	255	301
99,7	1,2	0,9	3,9	5,5	5,1	6	12	11	23	92	88	126	252	244	263
w odwrotnej kol.	1,1	0,8	3,5	4,8	4,5	7	10	9	14	81	78	91	230	223	245

Dla każdego sortowania, korzystając ze średniego czasu sortowania dla danych tablic sporządzono wykresy. Kształt wykresów każdego z algorytmu odpowiada teoretycznej złożoności obliczeniowej.


Sortowanie szybkie - porównanie dla różnych przypadków wstępnej zawartości tablicy
Czas sortowania [ms]
1800
1600
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
700
800
900
1000
ilość elementów [tyś]
0%
25%
50%
75%
95%
99%
99,7%
w odwrotnej kolejności


Sortowanie szybkie - po odrzuceniu najgorszego przypadku
Czas sortowania [ms]
350
300
250
200
150
100
50
0
0
100
200
300
400
500
600
700
800
900
1000
ilość elementów [tyś]
0%
25%
75%
95%
99%
99,7%
w odwrotnej kolejności


Sortowanie przez scalanie- porównanie dla różnych przypadków wstępnej zawartości tablicy
Czas sortowania [ms]
300
250
200
150
100
50
0
0
100
200
300
400
500
600
700
800
900
1000
ilość elementów [tyś]
0%
25%
50%
75%
95%
99%
99,7%
W odwrotnej kolejności


Sortowanie introspektwne - porównanie dla różnych przypadków wstępnej zawartości tablicy
Czas sortowania [ms]
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000
ilość elementów [tyś]
0%
25%
50%
75%
95%
99%
99,7%
w odwrotnej kolejnośći

4. Wnioski

Krzywe zobrazowane na wykresach odpowiadają w przybliżeniu klasach złożoności obliczeniowej danego algorytmu. Wyniki jakie uzyskano mogą się różnić na odmiennych konfiguracjach sprzętowych, ponieważ szybkość obliczeń w dużej mierze zależy od posiadanego sprzętu. Zgodnie z intuicją czas sortowania malał dla zwiększającej się liczby elementów wstępnie posortowanych dla każdego algorytmu. Sortowanie danych uporządkowanych w odwrotnej kolejności nie miało szczególnego wpływu na badane algorytmy i wykonywało się praktycznie w najkrótszym czasie.

Przeprowadzona analiza 3 algorytmów sortowania należących do klasy szybkich algorytmów pozwala stwierdzić, że wybór odpowiedniego sposobu sortowania zależy od ilości danych jakie posiadamy oraz ich uporządkowania. Dla dużej liczby danych (duża ilość elementów w tablicy) najefektywniejszym algorytmem okazało się sortowanie przez scalanie. Z przeprowadzonych testów wynika że ten sposób jest najbardziej uniwersalny i stabilny, więc sprawdzi się również w przypadku gdy nie wiemy z jaką ilością danych będziemy mieli do czynienia. Najbardziej niestabilne i nieprzewidywalne jest sortowanie szybkie, które w niekorzystnym przypadku ma złożoność obliczeniową podobną do najprostszych algorytmów typu sortowania przez wstawianie.

5. Literatura

Do analizy algorytmów korzystałem ze stron:

- www.geeksforgeeks.org
- https://eduinf.waw.pl/inf/alg/003_sort/mindex.php-wiki
- https://www.wikipedia.org/
- https://stackoverflow.com/