Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 1

Data: 01.04.2021

Dane studenta: Remigiusz Mielcarz 252887

Dane prowadzącego: Mgr inż. Marta Emirsajłow

Termin zajęć: Wtorek, 15^{15}-16^{55}

Termin oddania sprawozdania: 13.04.2021r.

1. Cel Projektu

Celem projektu jest porównananie działań sortowania w sferze czasowej oraz pamięciowej. Dzięki temu można wywnioskować, który algorytm jest najlepszy do naszych działań.
Opracowałem porównanie działań algorytmów poprzez:
kopcowanie, scalanie i shell.
Do przeprowadzenia testów efektywności generowałem 100 tablic rozmiarów: 10 000, 50 000, 100 000, 500 000, 1 000 000.
Rodzaje tablic:

- Wszystkie elementy tablicy losowe
- 25%, 50%, 75%, 95%, 99%, 99,7% poczatkowych elementów tablicy jest już posortowanych
- Wszystkie elementy tablicy juz posortowane ale w odwrotnej kolejnosci.

2. Opis algorytmów:

- **Kopcowanie** - Kopiec jest strukturą podobną do drzewa. Z sortowanej tablicy tworzymy kopiec binarny. W korzeniu kopca umieszczamy element największy (jeśli przyjmiemy sortowanie rosnące). Element ten zamieniamy z ostatnim elementem kopca. Elementy przenosimy na koniec tworząc część posortowaną i w kolejnych krokach nie ingerują już w skład kopca. Pozostały kopiec (bez elementów juz posortowanych) jest naprawiany (przywracanie własnosci kopca). Procedura jest powtarzana dopóki wszystkie elementy nie zostaną posortowane. Jego złożoność jest przewidywalna.
  Wstawienie jednego elementu wymaga maksymalnie log(n) operacji. Algorytm ten ma złożoność obliczeniową O(n log n) dla przypadku średniego jak i dla pesymistycznego.

- **Scalanie** - Opiera się na zasadzie „dziel i zwyciężaj". Główna zasada działania polega na rekurencyjnym dzieleniu tablicy na podtablice. Dzielenie kończymy, w którym, każda z podtablic w danej grupie jest tablicą jednoelementową. Łączymy je kolejno porównując wartości ich elementów. Dużą zaletą tego algorytmu jest możliwość zrównoleglenia - operacje sortowania można wykonywać na kilku wątkach lub nawet na wielu różnych maszynach w tym samym czasie. Złożoność obliczeniowa tego algorytmu dla przypadku średniego i pesymistycznego to O(n · log2n), a złożoność pamięciowa wynosi O(n).
- **Shell** – Jest to uogólnienie metody sortowania przez wstawianie, inaczje nazywny jest metodą malejących przyrostów. Bardzo ważnym elementem, który wpływa na efektywność sortowania metodą Shella jest odpowiednie dobranie ciągów odstępów.
  Na początku ustalamy pewną przerwę *p* pomiędzy elementami, które mają pozostać posortowane. Następnie w pętli dla każdego kolejnego elementu co *p*-ty element jest zamieniany jeśli nie spełniają warunku posortowania. Na koniec iteracji przerwa *p* jest zmniejszana według pewnej reguły. Działanie algorytmu kończy się, gdy następna wyliczona przerwa jest mniejsza od 1.
  Efektywność algorytmu sortowania metodą Shella zależy w dużym stopniu od ciągu przyjętych odstępów. Złożoność obliczeniowa sortowania przez wstawianie dla przypadku średniego to $O(n^2)$ tak samo jak dla przypadku pesymistycznego.

3. Wyniki testów

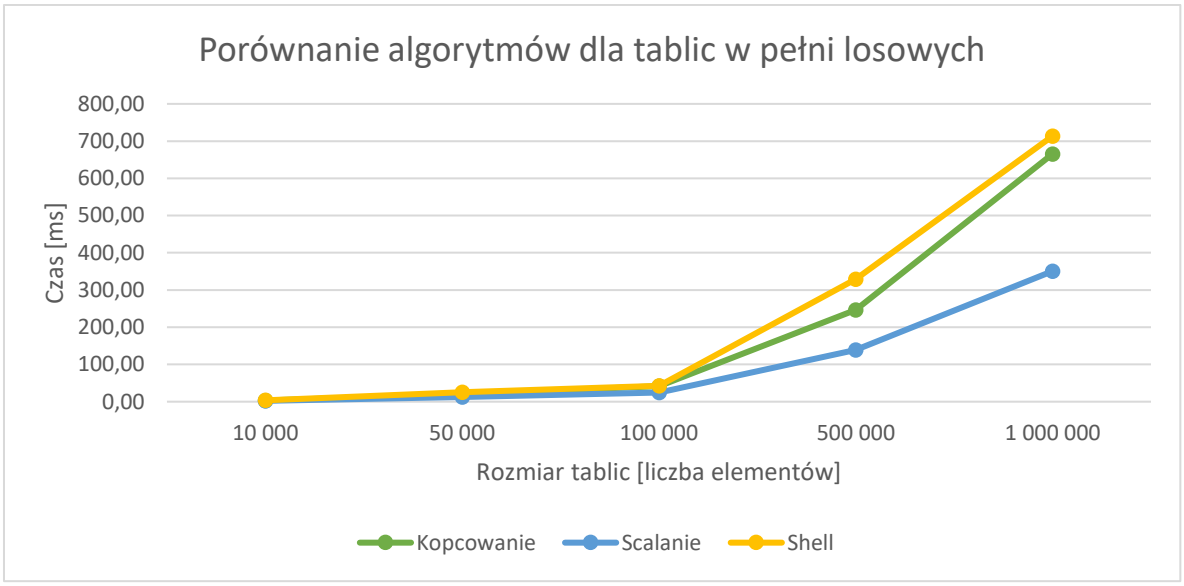
Wyniki testów są przeprowadzone w milisekundach, z dokładnością do 2 miejsc po przecinku.

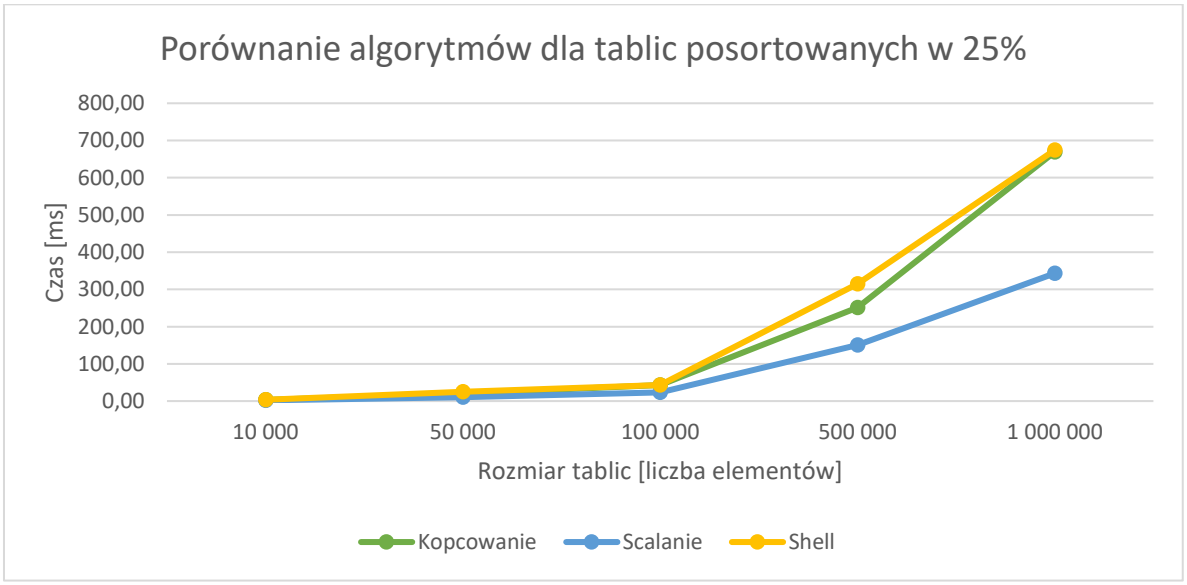
Kopcowanie					
	10 000	50 000	100 000	500 000	1 000 000
Losowe	3,36	19,50	41,50	246,57	665,56
Posortowane w 25%	3,23	19,99	42,87	251,23	669.43
Posortowane w 50%	3,00	20,51	43,22	261,18	654.57
Posortowane w 75%	3,58	20,82	44,52	317,13	612.92
Posortowane w 95%	3,70	21,86	47,24	257,96	569,43
Posortowane w 99%	3,62	21,79	45,80	257,46	477,80
Posortowane w 99,7%	3,66	21,81	45,27	283,17	486,35
Odwrotnie posortowane	3,14	19,29	40,57	285,10	537,38

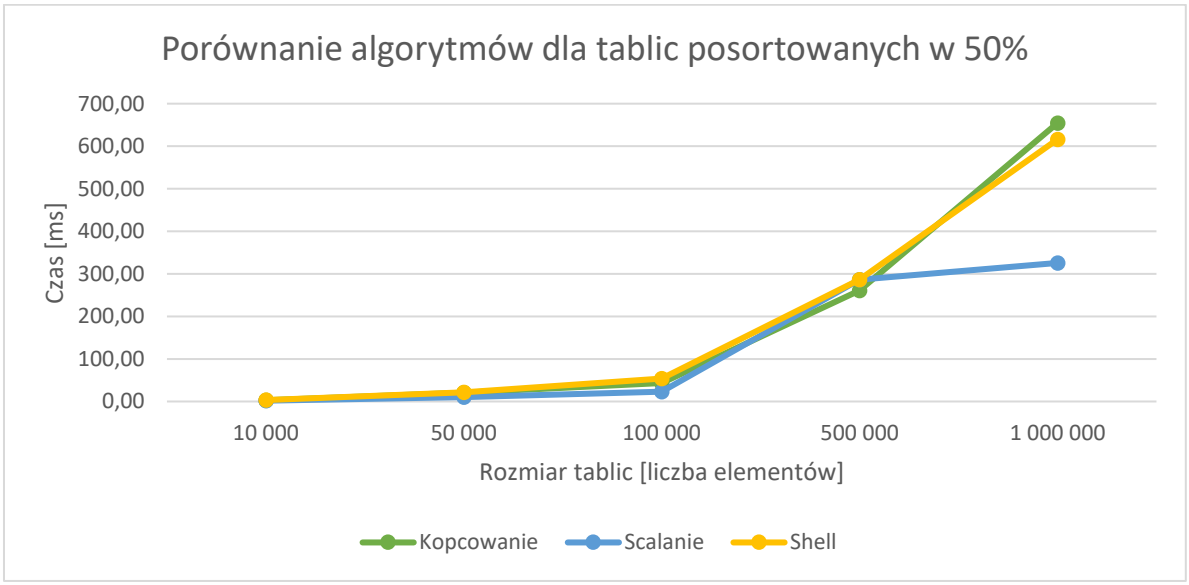
Scalanie					
	10 000	50 000	100 000	500 000	1 000 000
Losowe	2,28	12,53	24,89	138,90	350,64
Posortowane w 25%	1,98	10,71	23,16	150,74	343,09
Posortowane w 50%	1,90	10,55	22,67	286,63	325,68
Posortowane w 75%	1,94	10,24	20,81	152,82	306,75
Posortowane w 95%	1,54	9,88	20,21	139,68	293,55
Posortowane w 99%	1,88	9,19	20,25	137,04	290,72
Posortowane w 99,7%	1,84	9,75	19,75	138,38	297,56
Odwrotnie posortowane	2,26	10,29	20,89	136,66	283,42

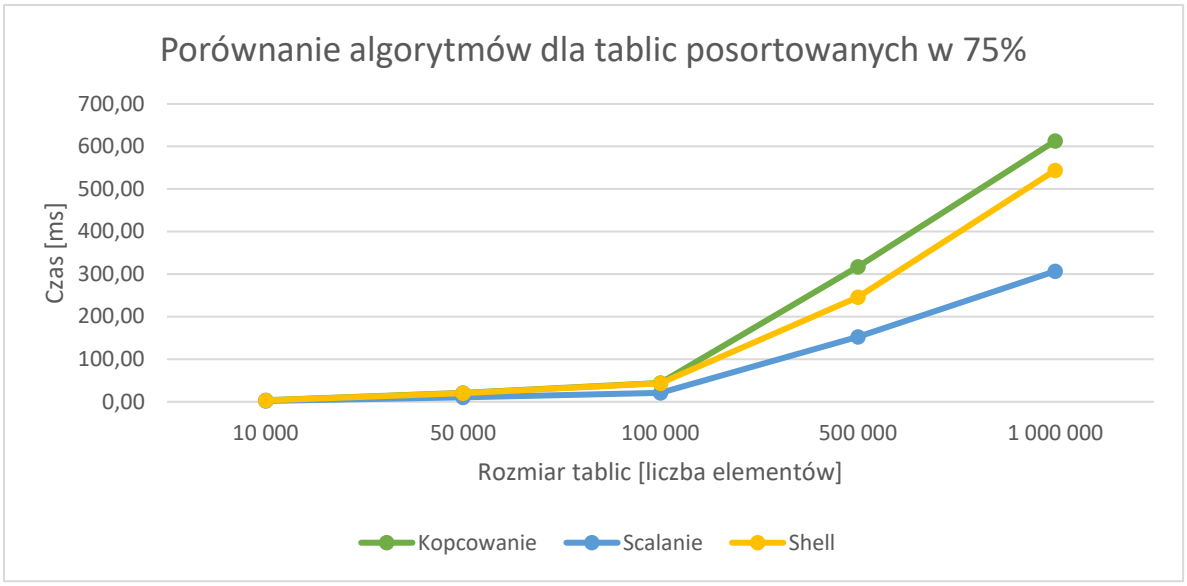
Shell					
	10 000	50 000	100 000	500 000	1 000 000
Losowe	3,61	25,43	42,72	328,80	713,61
Posortowane w 25%	3,62	24,66	43,02	314,58	674,37
Posortowane w 50%	3,36	21,90	53,95	286,34	616,19
Posortowane w 75%	3,17	20,48	43,47	246,18	543,96
Posortowane w 95%	2,52	15,50	34,74	191,62	418,55
Posortowane w 99%	1,85	12,37	26,45	153,78	298,42
Posortowane w 99,7%	1,85	10,02	25,80	147,63	293,67
Odwrotnie posortowane	0,78	3,99	9,72	53,48	105,15

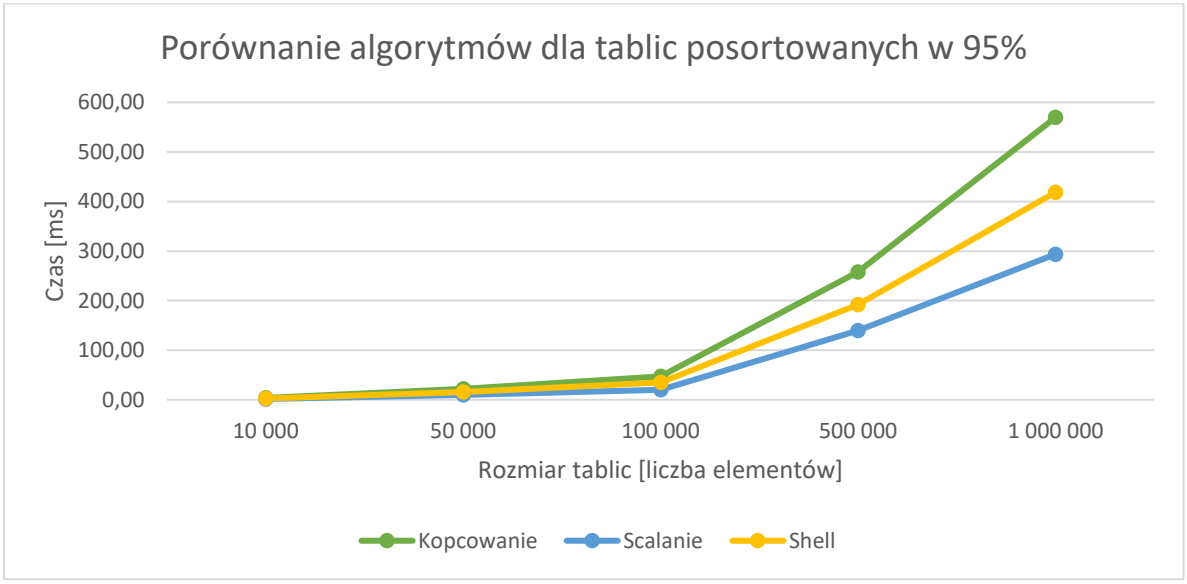
4. Wykresy

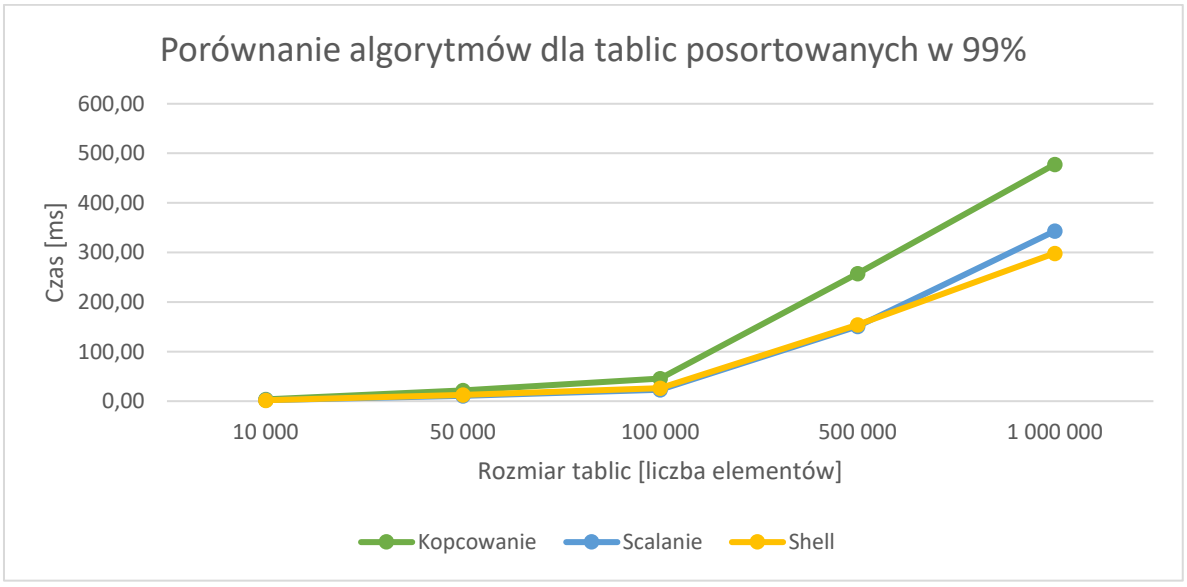

Porównanie algorytmów dla tablic w pełni losowych
800,00
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell

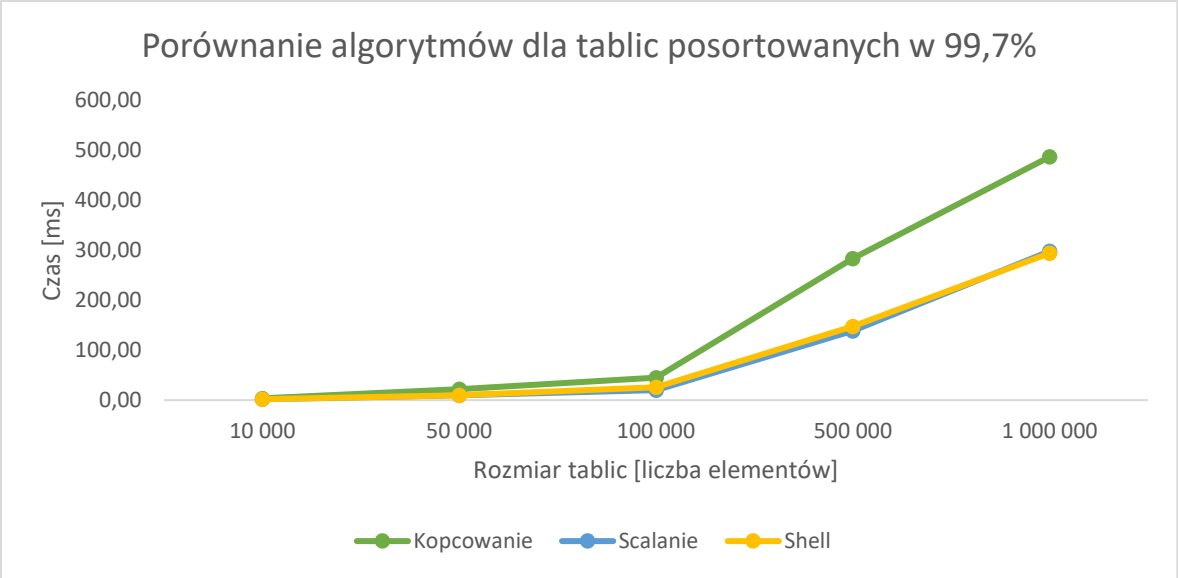

Porównanie algorytmów dla tablic posortowanych w 25%
800,00
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell

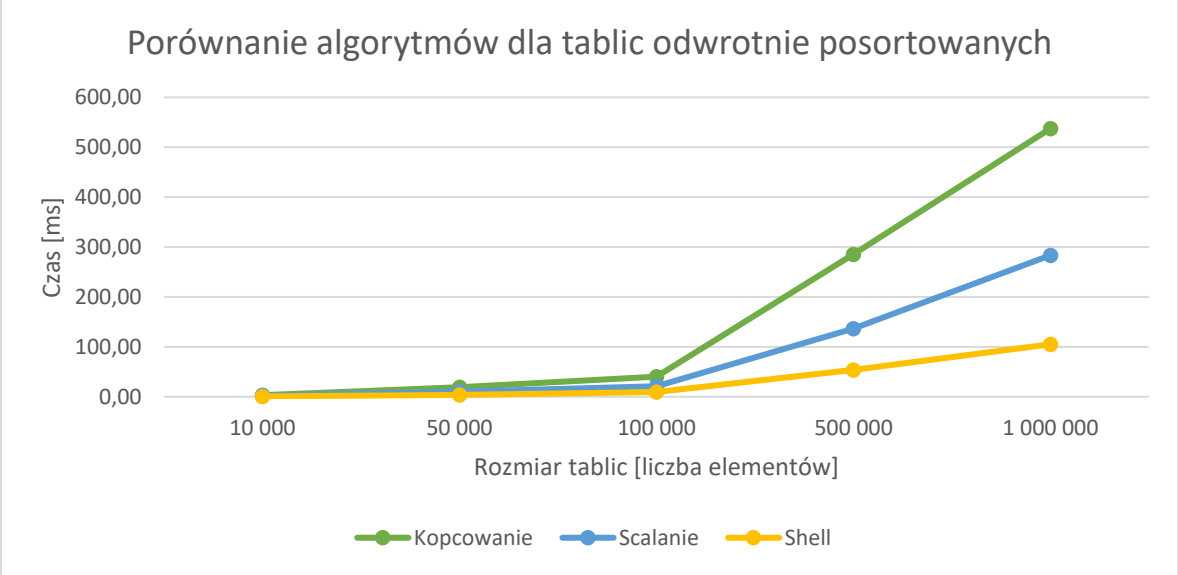

Porównanie algorytmów dla tablic posortowanych w 50%
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell


Porównanie algorytmów dla tablic posortowanych w 75%
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell


Porównanie algorytmów dla tablic posortowanych w 95%
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell


Porównanie algorytmów dla tablic posortowanych w 99%
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Czas [ms]
10 000
50 000
100 000
500 000
1 000 000
Rozmiar tablic [liczba elementów]
Kopcowanie
Scalanie
Shell





5. Wnioski

- Najwolniejszy algorytm sortowania jest poprzez kopcowanie. Wynika to z tego, że jego złożoność czasowa wynosi O(n log n), to oznacza, że czas wykonania algorytmu jest logarytmiczno-liniowy.
- Dla każdego z algorytmów przy tym samym rodzaju tablicy sortowanie z największą ilością elementów zajęło najwięcej czasu, ponieważ program musiał wykonać najwięcej linii kodu.

- Sortowanie Shella jest rozszerzeniem sortowania przez wstawianie. Proces sortowania jest znacznie przyśpieszony dzięki możliwości wymiany odległych od siebie elementów.
- Algorytm sortowania przez kopcowanie z częściowo posortowanymi tablicami najlepiej sobie radzi. Im wyższy procent posortowania, tym program jest szybszy. Wynika to ze sposobu obliczania, nie musi zamieniać kolejnością wielu elementów w kopcu.
- Sortowanie przez scalanie działa szybciej dla częsciowo posortowanych tablic, jednak ta zmiana nie jest aż tak duża jak w przypadku sortowania przez kopcowanie.