Projektowanie algorytmów i metody sztucznej inteligencji

25.03.2020r.

Mateusz Tkacz

Prowadzący : dr inż. Łukasz Jeleń

Czwartek 9:15

1. Wprowadzenie
   Algorytmy sortujące są wykorzystywane w celu porządkowania różnych danych. Różnią się one między sobą między innymi złożonością obliczeniową. Często proste algorytmy sortujące mają większą złożoność obliczeniową niż inne bardziej skomplikowane. To jaki algorytm warto będzie wykorzystać zależy od tego jak dużo danych będzie trzeba uporządkować. Projekt polegał na sprawdzeniu wpływu różnych czynników na algorytmy sortujące. Zbadany został wpływ uporządkowania elementów oraz ich ilość na czasy sortowania szybkiego, introspektywnego oraz sortowania przez scalanie.

2. Opis algorytmów
   1) Sortowanie szybkie
      Algorytm ten wykorzystuje technikę „dziel i rządź”. Spośród zbioru elementów wybierany jest jeden, tzw. piwot. Następnie reszta elementów jest porządkowana na 2 grupy – większe i mniejsze od piwota. Potem algorytm wywoływany jest rekurencyjnie dla dwóch podzbiorów, aż do momentu, gdy cały zbiór zostanie posortowany. Algorytm sortowania szybkiego zazwyczaj posiada złożoność obliczeniową O(nlogn). Jednak w wypadku, gdy jako piwot zostanie wybrany np. najmniejszy element zbioru, jego złożoność może się zdegradować do $O(n^2)$. Uniknąć tego można stosując różne metody wyboru piwota, gwarantujące, że nie będzie on skrajnym elementem, np. mediana z trzech.
   2) Sortowanie przez scalanie
      Algorytm ten również wykorzystuje technikę „dziel i rządź”. Polega on na tym, że dany zbiór dzieli na dwa podzbiory, które znów są dzielone na dwa, aż do momentu uzyskania zbiorów jednoelementowych. Następnie zbiory te są ze sobą łączone w uporządkowany sposób. Algorytm ten posiada złożoność obliczeniową O(nlogn) ze względu na to, że zbiór jest dzielony rekurencyjnie na 2 podzbiory i tworzy wtedy „drzewo”, jego wysokość zależy logarytmicznie od ilości elementów.
   3) Sortowanie introspektywne
      Jest to algorytm hybrydowy łączący sortowanie szybkie i sortowanie przez kopcowanie. Jest stosowany ze względu na to, że w pewnych przypadkach dla sortowania szybkiego występuje duża liczba rekurencyjnych wywołań, zwłaszcza dla małych zbiorów. Dlatego, gdy zbiory będą mniejsze niż wyznaczony próg, zostaną one posortowane przez kopcowanie, które posiada stałą złożoność obliczeniową O(nlogn).

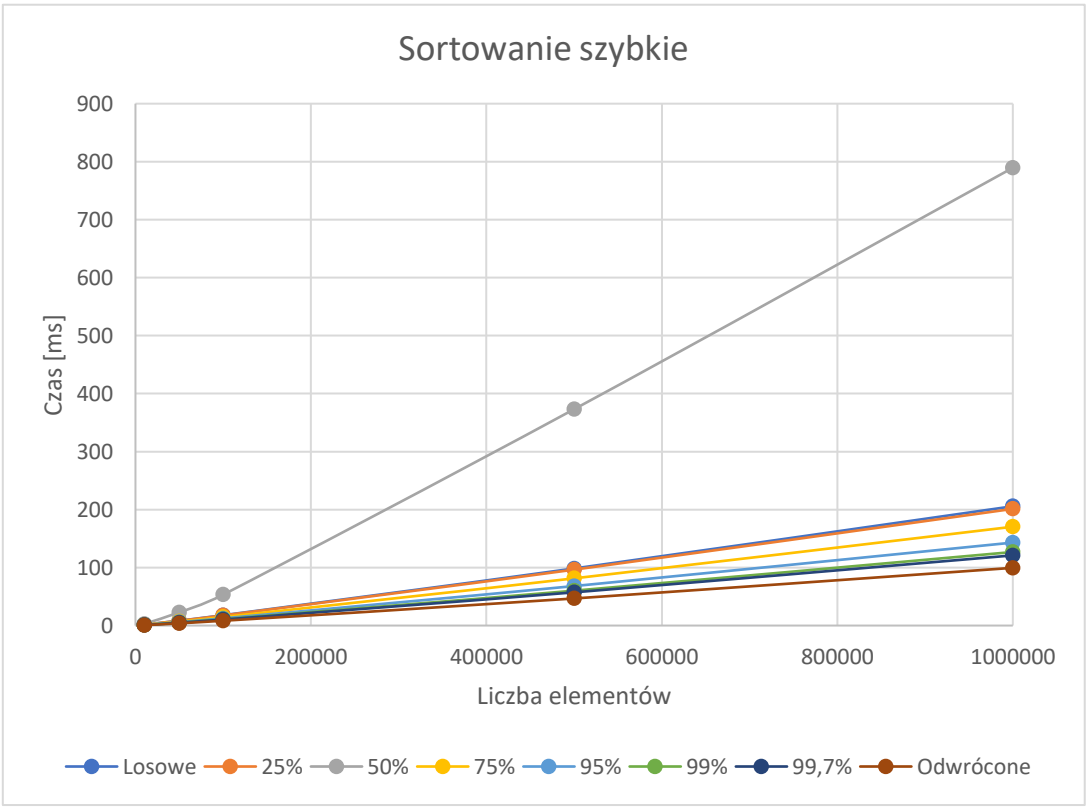
3. Przebieg doświadczeń

Doświadczenie przeprowadzane było na tablicach z losowo generowanymi elementami, które następnie były sortowane w różnym stopniu. Mierzony był czas wykonywania algorytmu dla 100 tablic o danym porządku elementów.

1) Sortowanie szybkie

Dla algorytmu sortowania szybkiego wzrost uporządkowania elementów w tablicy jest korzystny pod względem czasu sortowania. Jedyny wyjątek stanowi zbiór posortowany w 50%, dla którego czas działania znacznie się wydłuża. Najkrótszy czas wystąpił dla zbioru uporządkowanego w odwróconej kolejności.

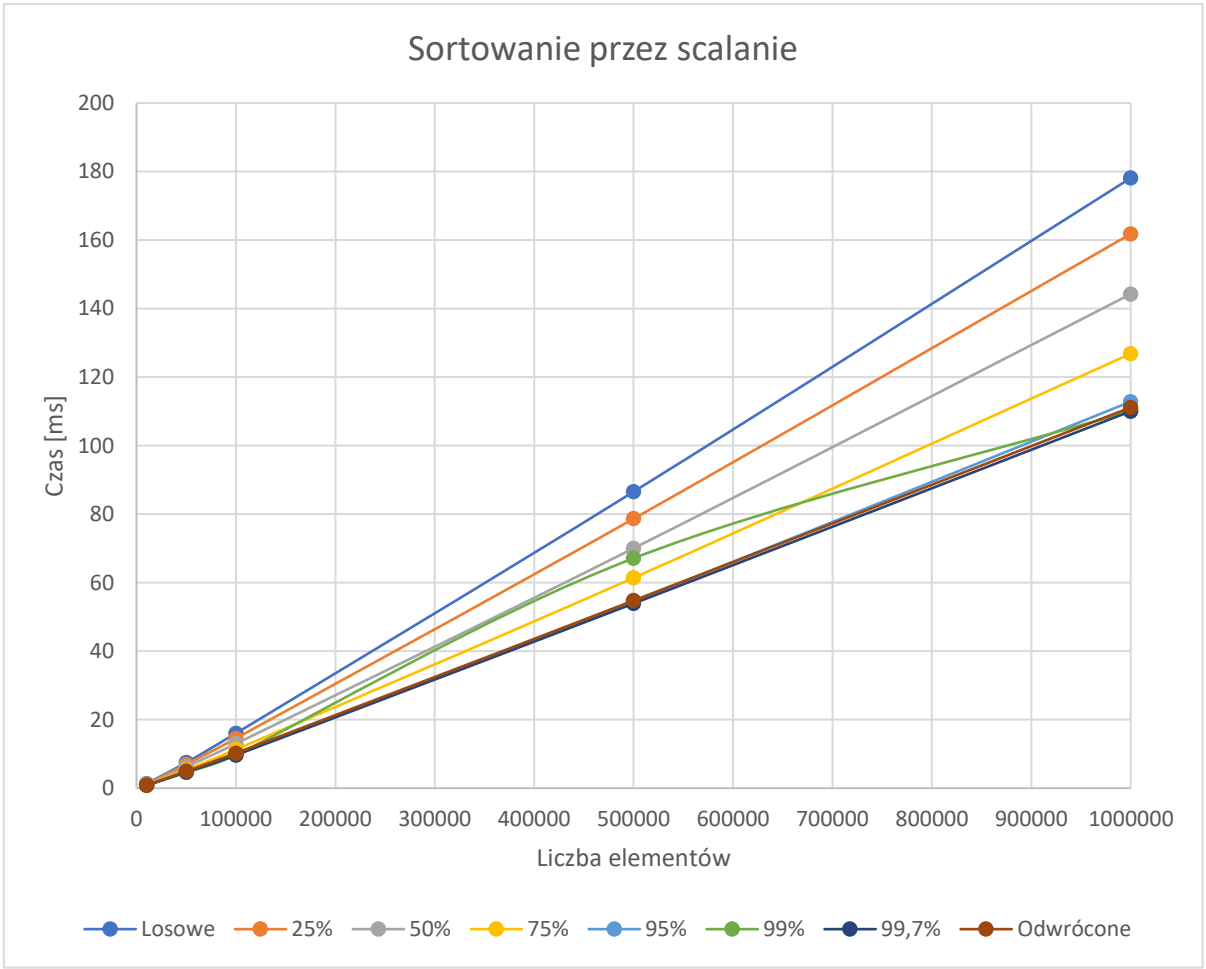
Sortowanie szybkie								
Liczba elementów	Porządek elementów w tablicy							
	Losowy	25%	50%	75%	95%	99%	99,70%	Odwrócony
10000	1,520	1,490	2,780	1,300	1,120	0,900	0,810	0,650
50000	8,480	8,310	22,860	7,010	5,910	5,200	4,840	3,850
100000	17,840	17,450	53,280	14,710	12,360	11,060	10,340	8,300
500000	98,430	96,420	373,090	81,390	68,110	60,120	57,290	46,710
1000000	205,670	201,230	789,400	170,420	143,060	126,820	120,990	99,490



2) Sortowanie przez scalanie

Czas sortowania przez scalanie zmniejszał się wraz ze wzrostem uporządkowaniu zbioru sortowanego. Jednak dla zbioru uporządkowanego w odwrotnej kolejności nie był najszybszy.

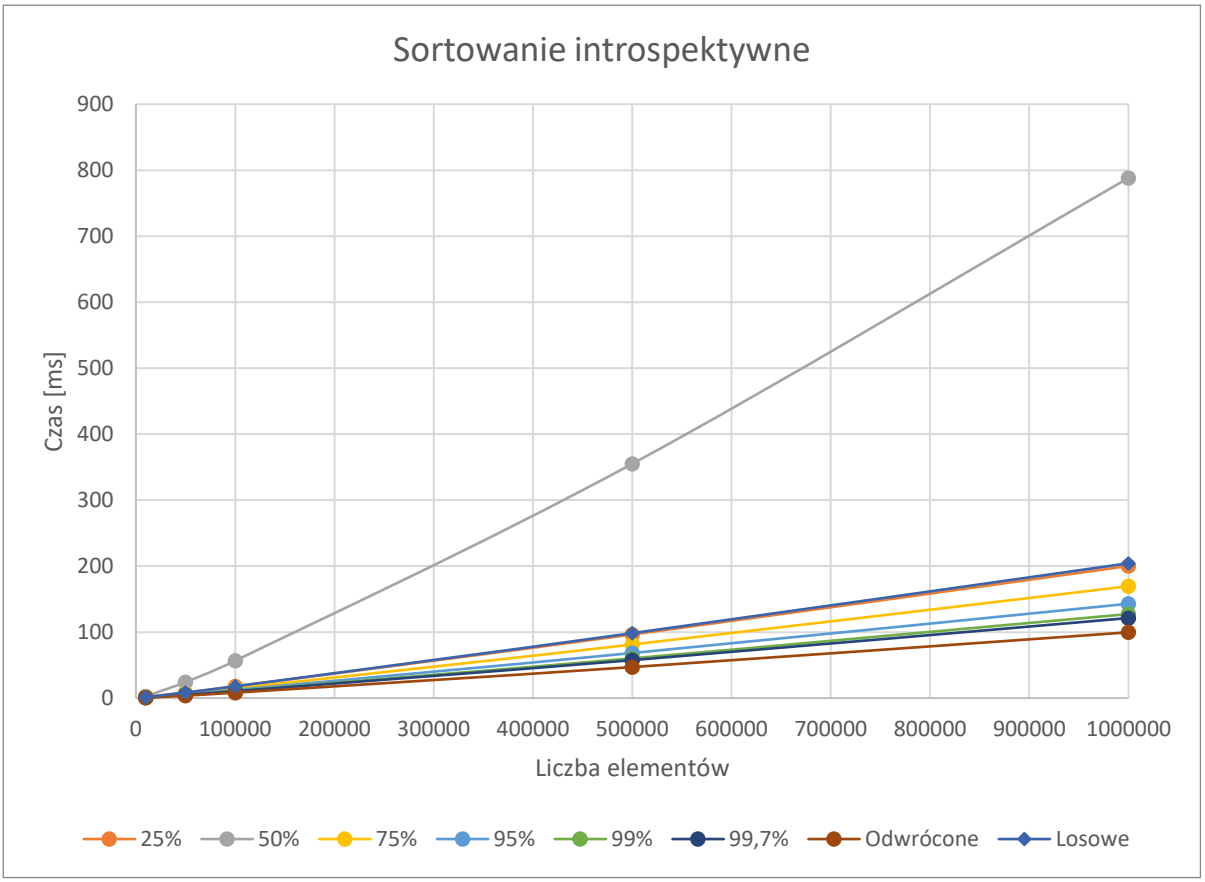
Sortowanie przez scalanie								
Liczba elementów	Porządek elementów w tablicy							
	Losowy	25%	50%	75%	95%	99%	99,70%	Odwrócony
10000	1,330	1,220	1,120	0,960	0,900	0,840	0,830	0,850
50000	7,460	6,970	6,260	5,370	4,800	4,640	4,660	4,960
100000	16,010	14,550	12,990	11,290	10,010	9,710	9,660	10,240
500000	86,550	78,660	70,020	61,410	54,540	67,120	53,910	54,770
1000000	178,100	161,790	144,230	126,830	112,830	110,160	110,030	111,150



3) Sortowanie introspektywne

Podobnie jak w przypadku sortowania szybkiego, znacznie dłuższy czas od pozostałych wystąpił dla zbioru uporządkowanego w 50%.

Sortowanie introspektywne								
Liczba elementów	Porządek elementów w tablicy							
	Losowy	25%	50%	75%	95%	99%	99,70%	Odwrócony
10000	1,510	1,430	2,720	1,230	1,050	0,930	0,800	0,640
50000	8,510	8,300	24,320	6,960	5,900	5,030	4,850	3,720
100000	17,780	17,710	56,530	14,650	12,350	10,970	10,390	8,240
500000	98,380	96,360	354,980	81,190	68,280	60,230	57,490	46,980
1000000	204,290	200,200	788,200	169,470	142,970	127,330	121,290	99,640



4. Wnioski

   Na powyższych wykresach widać stosunkowo łagodny wzrost czasu w zależności od wzrostu elementów, co daje podstawy by twierdzić, że implementacja algorytmów jest względnie optymalna. Dla algorytmu sortowania szybkiego i introspektywnego przy zbiorze uporządkowanym w 50% czas sortowania jest znacznie większy niż w pozostałych przypadkach . Może to oznaczać, że w tym układzie algorytm wykonuje dużo wywołań rekurencyjnych, co powoduje wydłużenie czasu sortowania.

5. Literatura

   1) https://en.wikipedia.org/wiki/Quicksort
   2) http://informatyka.2ap.pl/ftp/3d/algorytmy/podręcznik_algorytmy.pdf?fbclid=IwAR0Tdj-AeezBVyId3q57SaCimltnuNdEU4rd9SP0VI1UATRrGAGe7ClhLR4
   3) https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie