Sprawozdanie 1

"Projektowanie algorytmów i metod sztucznej inteligencji"

27 marca 2019

Temat projektu : Algorytmy sortowania oraz ich złożoność obliczeniowa.
Autor : Przemysław Widz
Termin zajęć : ŚRODA 7:30 - 9:00
Prowadzący : Dr inż. Łukasz Jeleń

1 Wstęp teoretyczny

1.1 Sortowanie przez scalanie

Sortowanie przez scalanie należy do algorytmów , które wykorzystują metodę podziału głównego problemu na mniejsze podproblemy. Metoda ta nosi nazwę „dziel i zwyciężaj". Złożoność algorytmu wynosi O(n log n) , a więc jest on znacznie wydajniejszy niż sortowania takie jak sortowanie bąbelkowe , czy przez selekcję, gdzie złożoność jest kwadratowa.

Działanie tego algorytmu można przedstawić w trzech krokach :

1. podziel zestaw danych na dwie równe części
2. zastosuj sortowanie przez scalanie dla każdej z nich oddzielnie, chyba, że pozostał już tylko jeden element
3. połącz posortowane podciągi w jeden ciąg posortowany

Zalety tego algorytmu to :

- prostota implementacji
- wydajność
- stabilność

Wady tego algorytmu to :

- podczas sortowania potrzebujemy dodatkowy obszar pamięci przechowujący kopie podtablic do scalania

1.2 Sortowanie szybkie

Sortowanie szybkie należy również do grupy algorytmów „dziel i zwyciężaj". Według ustalonego schematu, wybierany jest jeden element w sortowanej tablicy, który będziemy nazywać pivotem . Pivot może być dowolnym elementem tablicy. Następnie ustawiamy elementy nie większe na lewo tej wartości , natomiast nie mniejsze na prawo. W taki oto sposób otrzymujemy dwie części tablicy (niekoniecznie równe) , gdzie w pierwszej części znajdują się elementy nie większe od drugiej. Następnie każdą z tych podtablic sortujemy osobno według tego samego schematu. W przygotowanym przeze mnie programie , pivot wybieramy jako element środkowy tablicy.

Do zalet tego algorytmu neleżą :

- logarytmiczna złożoność czasowa (dla przypadku optymistycznego)
- algorytm nie potrzebuje dodatkowej tablicy do jego wykonania
- łatwość w implementacji, dobrze współpracuje z różnymi typami danych.

Wady algorytmu sortowania szybkiego to :

- w pesymistycznym przypadku złożoność obliczeniowa algorytmu może być kwadratowa, jest niestabilny, jest wrażliwy na błędy w implementacji.

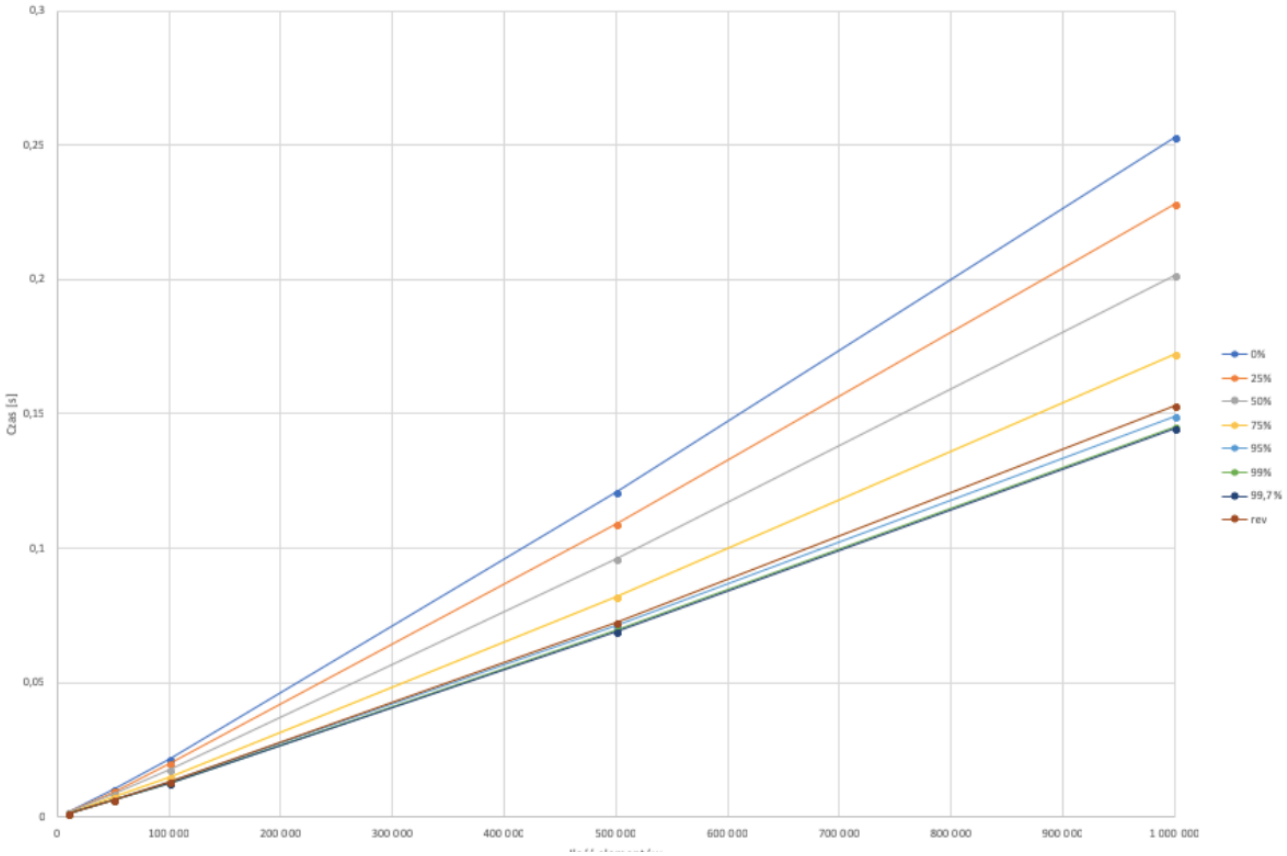
1.3 Sortowanie introspektywne

Jest to odmiana sortowania hybrydowego, w której wyeliminowany został problem kwadratowej złożoności obliczeniowej, występującej w najgorszym przypadku algorytmu sortowania szybkiego. Metoda ta polega na kontrolowaniu głębokości wywoływania rekurencyjnego dla algorytmu sortowania szybkiego. W przypadku, gdy pewna wartość M, określona jako 2·log2n , obliczona na początku działania programu oraz zmniejszana z każdym ponownym wywołaniem rekurencyjnym algorytmu sortowania szybkiego osiągnie zero, dla podproblemu którym obecnie się zajmujemy wywoływana jest procedura Sortowania przez kopcowanie, które jest traktowane jako sortowanie pomocnicze. Natomiast na końcu działania algorytmu, wywoływany jest inny algorytm sortowania np. sortowanie przez wstawianie dla posortowanej już wstępnie tablicy.

2 Sortowanie przez scalanie (ang."merge sort")

		10k	50k	100k	500k	1000k
0.00%	średnia[s]	0.00190906	0.0102914	0.0216312	0.120768	0.25311
25.00%		0.00172856	0.00933648	0.0195873	0.109053	0.228369
50.00%		0.00145941	0.0082779	0.0174004	0.0961001	0.201451
75.00%		0.00130812	0.00709999	0.0148958	0.0820637	0.172414
95.00%		0.00111349	0.00616719	0.0129762	0.0712569	0.149239
99.00%		0.00108168	0.00604304	0.012601	0.069535	0.144891
99.70%		0.0010739	0.00599222	0.0125031	0.0689943	0.144553
odwrócona		0.00110179	0.00617615	0.0129275	0.0725054	0.152771

Tabelka 1. Tabelka dla sortowania przez scalanie. Czas średniego wykonania algorytmu dla jednej tablicy podano w sekundach.



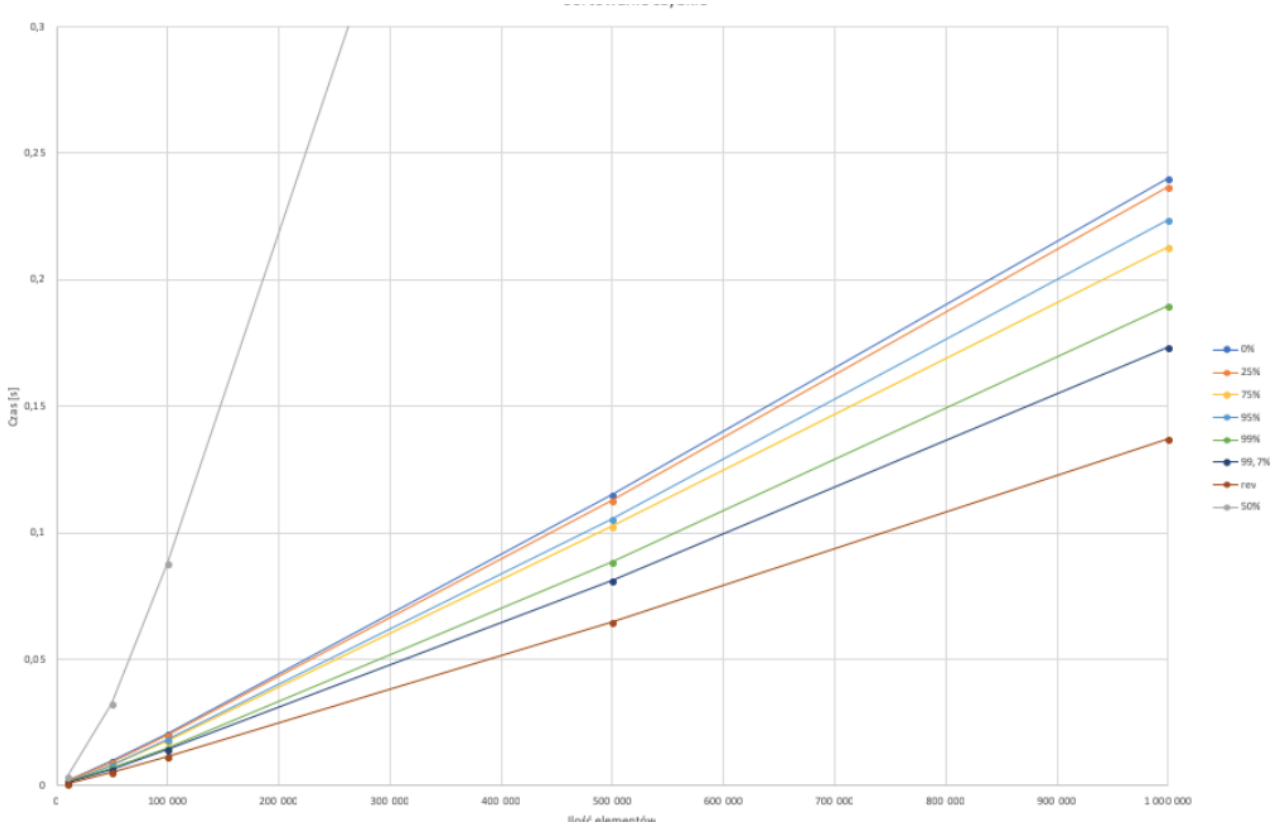
Rysunek 1: Wykres zależności czasu wykonania algorytmu sortowania przez scalanie .

Uwaga 1. Kształt wykresu potwierdza złożoność obliczeniową algorytmu sortowania przez scalanie na poziomie O(n log n).

3 Sortowanie szybkie (ang. "quick sort")

		10k	50k	100k	500k	1000k
0.00%	średnia [s]	0.0017177	0.00957844	0.020469	0.114897	0.239997
25.00%		0.00161486	0.00943185	0.0201924	0.112751	0.236737
50.00%		0.00385232	0.0326187	0.0878564	0.60906	1.97617
75.00%		0.00142941	0.00827093	0.0177213	0.102323	0.213061
95.00%		0.00136369	0.00829922	0.0180548	0.105424	0.223621
99.00%		0.00119723	0.0069821	0.0150302	0.0886362	0.189668
99.70%		0.00110538	0.00658375	0.0141363	0.0812416	0.173537
odwrócona		0.00095929	0.00530355	0.0112565	0.0644402	0.137215

Tabelka 2. Tabelka dla sortowania szybkiego. Czas średniego wykonania algorytmu dla jednej tablicy podano w sekundach.



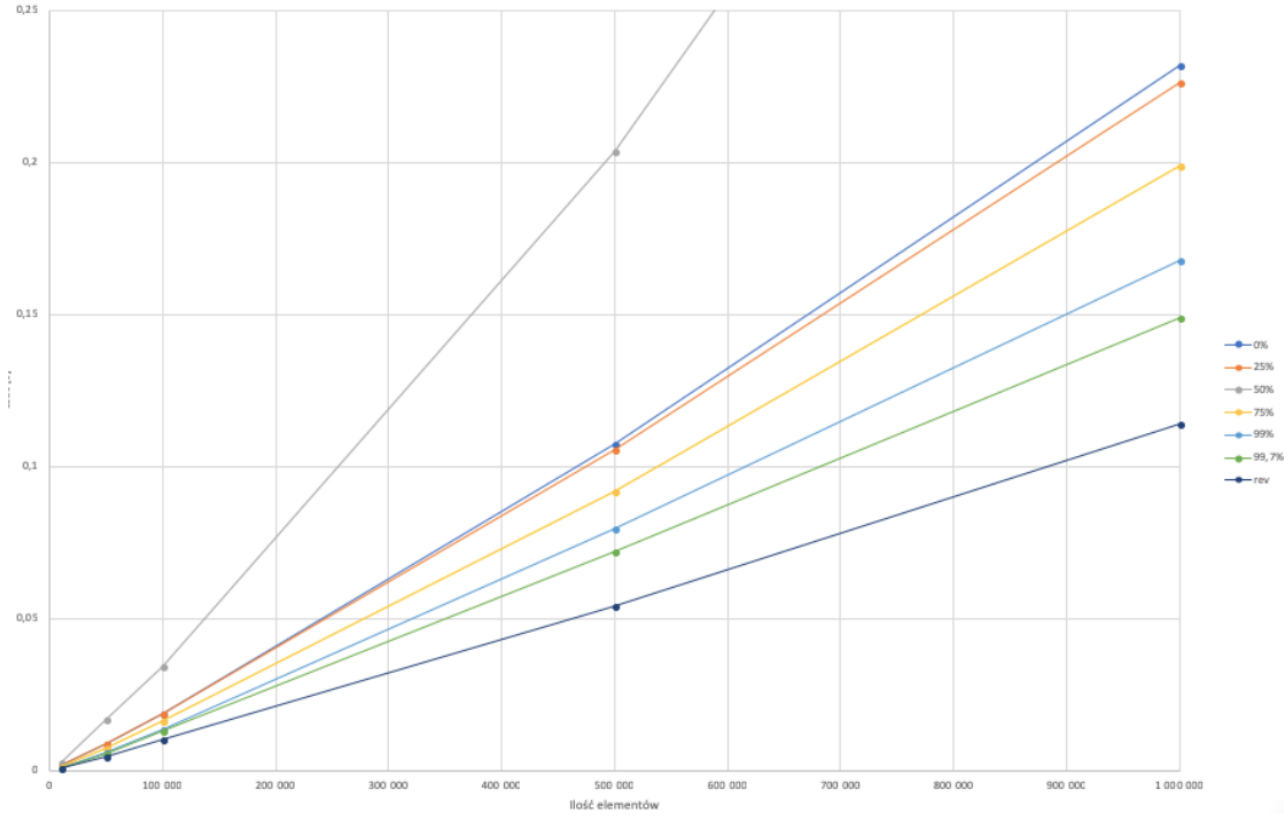
Rysunek 2: Wykres zależności czasu wykonania algorytmu sortowania szybkiego .

Uwaga 2. Kształt wykresu potwierdza złożoność obliczeniową algorytmu sortowania szybkiego na poziomie O(n log n) (w przypadku optymistycznym). Natomiast dla przypadku , gdzie 50 % tablicy jest już posortowane, możemy zauważyć, że algorytm ten działa o wiele wolniej , niż dla innych przypadków i jego złożoność obliczeniowa zbliża się wtedy do złożoności O(n^2.)

4 Sortowanie introspektywne (ang. „introsort”)

		10k	50k	100k	500k	1000k
0.00%	średnia [s]	0.00149222	0.00876674	0.0189131	0.107547	0.231915
25.00%		0.00151146	0.00861131	0.0184719	0.105661	0.226278
50.00%		0.002772	0.0167147	0.0343262	0.203514	0.465776
75.00%		0.00123841	0.00746054	0.0162564	0.0917897	0.199085
95.00%		0.00115369	0.00731686	0.0159761	0.0972085	0.206681
99.00%		0.00098006	0.00589442	0.0133023	0.0796934	0.167721
99.70%		0.00089156	0.00548398	0.0131802	0.0721793	0.149058
odwrócona		0.00076833	0.00446418	0.0100442	0.0540535	0.113941

Tabelka 3. Tabelka dla sortowania introspektywnego. Czas średniego wykonania algorytmu dla jednej tablicy podano w sekundach.



Rysunek 3: Wykres zależności czasu wykonania algorytmu sortowania introspektywnego .

Uwaga 3. Kształt wykresu potwierdza złożoność obliczeniową algorytmu sortowania introspektywnego na poziomie O(n log n) . Dla przypadku , gdzie 50% tablicy jest już posortowane, możemy zauważyć, że algorytm ten działa o wiele wolniej , niż dla innych przypadków , natomiast i tak otrzymujemy na wykresie linię zbliżoną bardziej do linii prostej , a nie paraboli , wskazującą na złożoność obliczeniową na poziomie bliższym O(n log n), a nie O(n^2).

5 Analiza danych i wnioski

1. Zaimplementowany algorytm sortowania przez scalanie jest najstabilniejszy oraz zachowuje się najbardziej „intuicyjnie" , tzn. dla tablic już wstępnie posortowanych , w zależności od stopnia ich posortowania , trwa coraz krócej. Natomiast dla tablic wstępnie nieposortowanych (tj. przypadek 0% , niezależnie od rozmiaru tablicy , sortowanie trwa dłużej niż w przypadku sortowania szybkiego , czy sortowania introspektywnego).

2. Algorytm sortowania szybkiego nie działa już tak intuicyjnie jak algorytm sortowania przez scalanie. Najgorszy przypadek tego algorytmu otrzymujemy dla sytuacji , gdzie tablica posortowana jest wstępnie w 50 %. Odbiega on znacznie od pozostałych przypadków, tzn. trwa o wiele dłużej od pozostałych. Wynika to najprawdopodobniej z metody wyboru pivota , jako elementu środkowego tablicy.

3. Algorytm sortowania introspektywnego , jak można się było spodziewać, kontrolując głębokość wywołań rekurencyjnych quickSorta , jest od niego szybszy i usprawnia tym samym jego działanie. Jednak ze względu na metodę wyboru pivota, używanej w sortowaniu szybkim , nadal przypadek dla tablicy wstępnie posortowanej w 50% jest najwolniejszy. Wykonuje się on jednak krócej niż dla algorytmu sortowania szybkiego, więc można uznać, że przypadek złożoności obliczeniowej na poziomie O(n^2) dla tego algorytmu został wyeliminowany (na wykresie otrzymujemy prostą o mniejszym nachyleniu, niż dla sortowania szybkiego).

6 Literatura

1. https://pl.wikipedia.org/wiki/Sortowanie$_p rzez_s calanie$
2. https://pl.wikipedia.org/wiki/Sortowanie$_s zybkie$
3. https://pl.wikipedia.org/wiki/Sortowanie$_i ntrospektywne$
4. https://www.youtube.com/watch?v=iJyUFvvdfUg
5. https://www.youtube.com/watch?v=82XxdhRCMbI
6. https://www.szkolnictwo.pl/szukaj,Sortowanie$_i ntrospektywneSortowanie_H ybrydowe$
7. http://www.algorytm.edu.pl/