Sprawozdanie z ćwiczenia laboratoryjnego
z
Projektowania Algorytmów i Metod Sztucznej Inteligencji

Sprawozdanie z 5-tego i 6-tego laboratorium

Sprawozdanie wykonał:

Rafał Januszewski

Data wykonania sprawozdania: 18.04.2016 r.
Data wykonywanego ćwiczenia: 04.04.2016 r.
Termin zajęć: poniedziałek 08:15

1. Wprowadzenie

Celem ćwiczenia laboratoryjnego zaimplementowanie i zbadanie efektywności trzech wybranych algorytmów sortowania. Do badania zostały wybrane następujące algorytmy sortowania: sortowanie przez scalanie, sortowanie szybkie (quicksort) oraz sortowanie Shella. Testy efektywności algorytmu polegały na wygenerowaniu stu tablic o rozmiarach 10000, 50000, 100 000, 500 000 i 1 000 000 zawierających liczby losowe typu całkowitoliczbowego, a następnie ich posortowaniu przy uwzględnieniu następujących przypadków: tablica wypełniona jest wartościami losowymi, pierwsze 25%, 50%, 75%, 99%, 99,7% elementów tablicy jest wstępnie posortowanych. Powyższe algorytmy sortowania były testowane na systemie operacyjnym Ubuntu 14.04 LTS 64-bit. Parametry sprzętowe były następujące: model procesora i częstotliwość taktowania: Intel Core i3 M380 2,53GHz, pamięć RAM 2,8GiB.

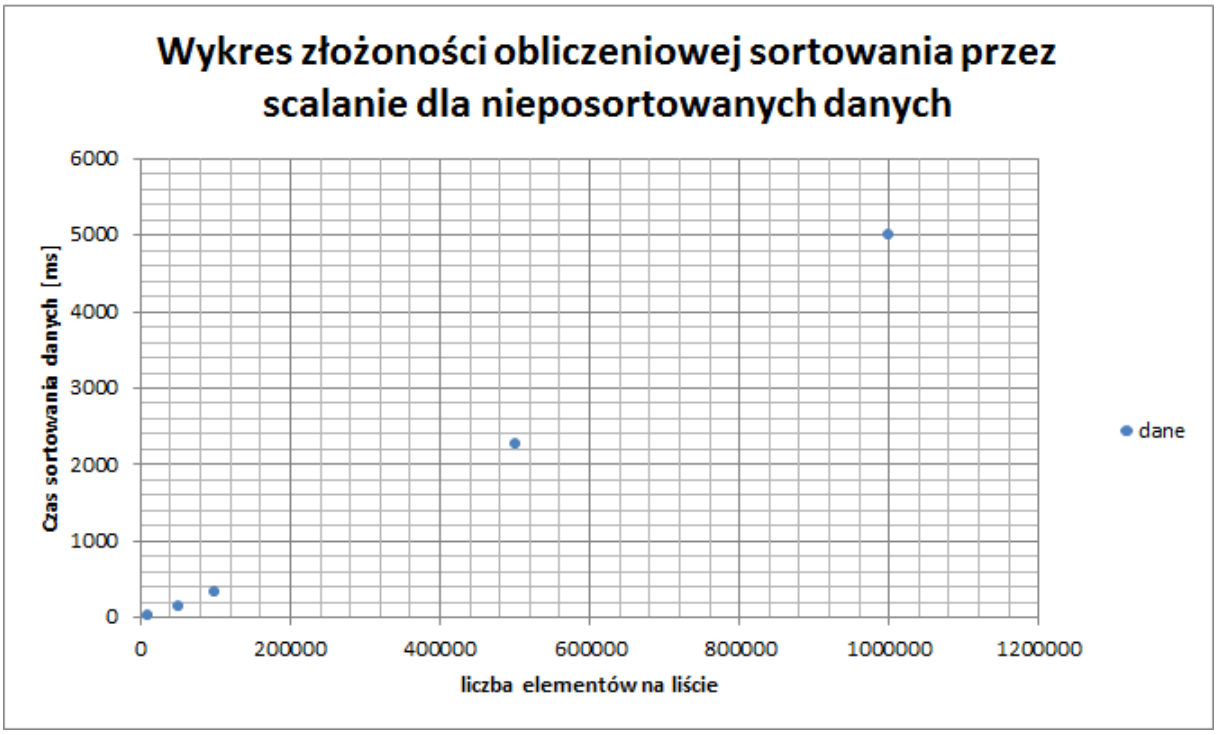
2. Badanie algorytmu sortowania przez scalanie

Sortowanie przez scalanie jest algorytmem opartym na zasadzie dziel i zwyciężaj, wywoływanym rekurencyjnie. Sortowanie przez scalanie odbywa się poprzez dzielenie ciągu danych na dwie równe części, wykonanie sortowania dla każdej z nich oddzielnie i połączeniu

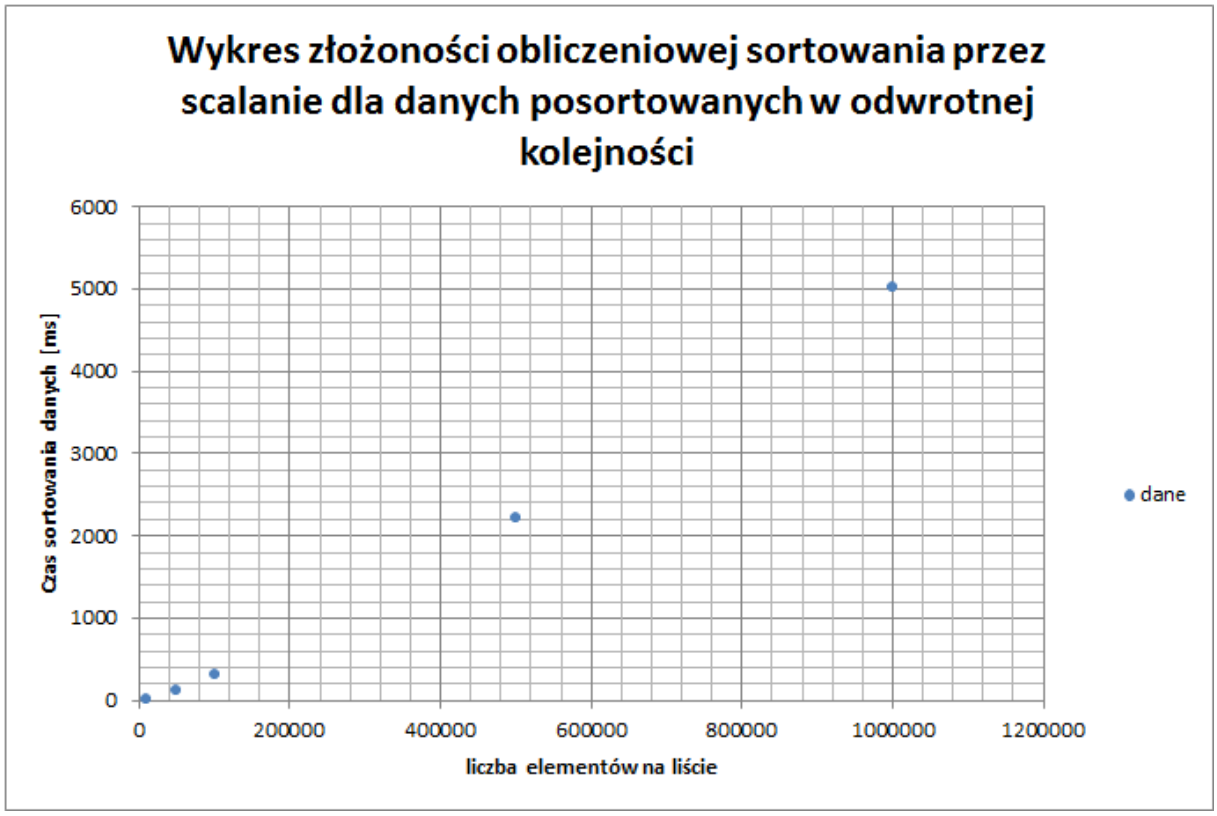
podciągów danych w jedno. Złożoność obliczeniowa średniego i najgorszego przypadku powinna wynosić $nlog(n)$.

Tabela 1. Badanie czasu sortowania danych

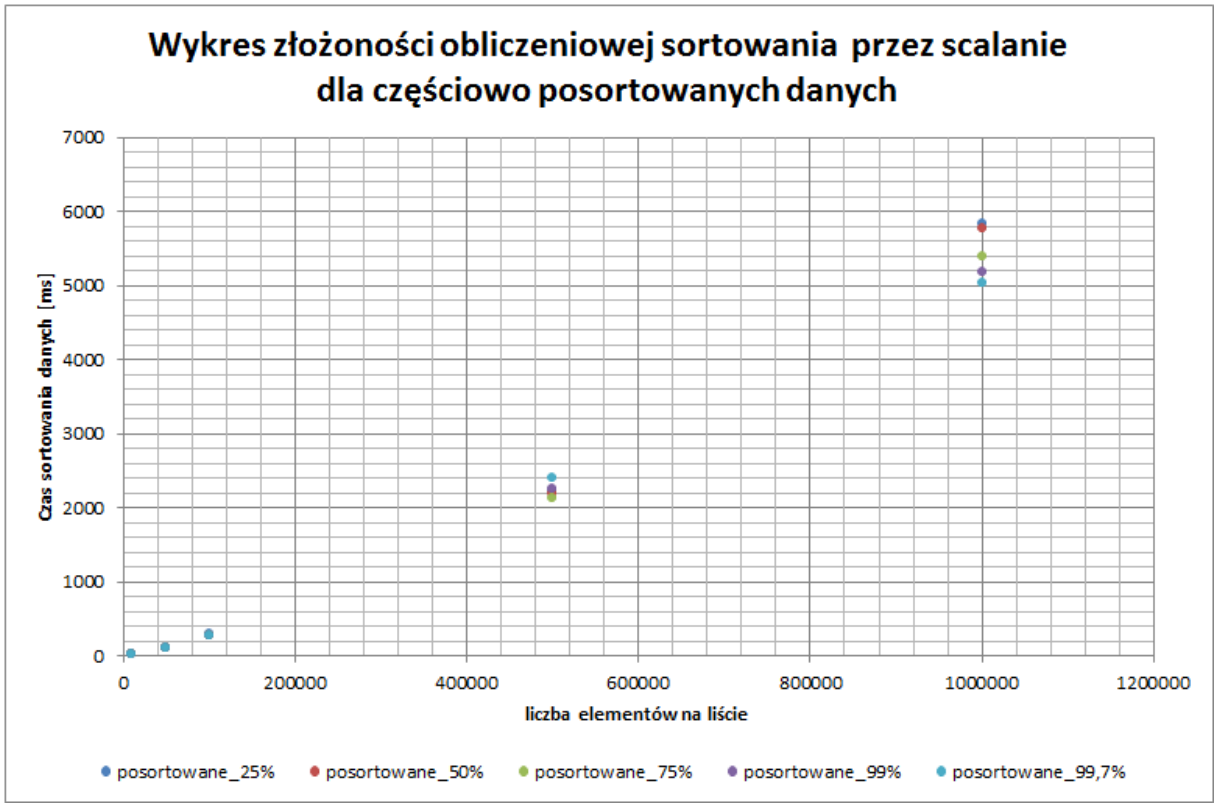
liczba elementów	czas sortowania [ms]						
	wszystkie elementy losowe	posortowana w odwrotnej kolejności	Posortowana część listy [%]				
			25	50	75	99	99,7
10000	20	18	19	19	19	18	18
50000	134	120	120	120	115	111	114
100000	339	309	296	285	277	271	269
500000	2257	2219	2214	2194	2128	2263	2408
1000000	5014	5027	5835	5784	5396	5189	5045



Rys.1. Wykres złożoności obliczeniowej sortowania przez scalanie dla nieuporządkowanych danych



Rys.2. Wykres złożoności obliczeniowej sortowania przez scalanie dla danych uporządkowanych w odwrotnej kolejności



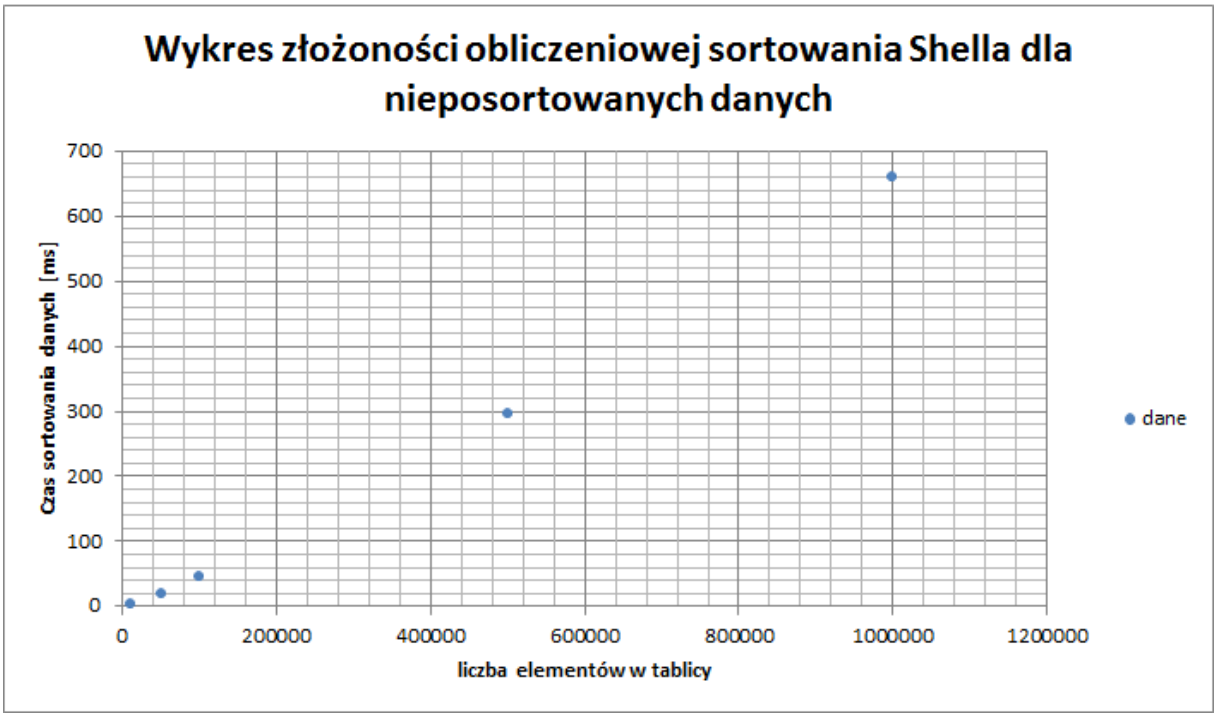
Rys.3. Wykres złożoności obliczeniowej sortowania przez scalanie dla danych częściowo posortowanych

3. Badanie algorytmu sortowania Shella

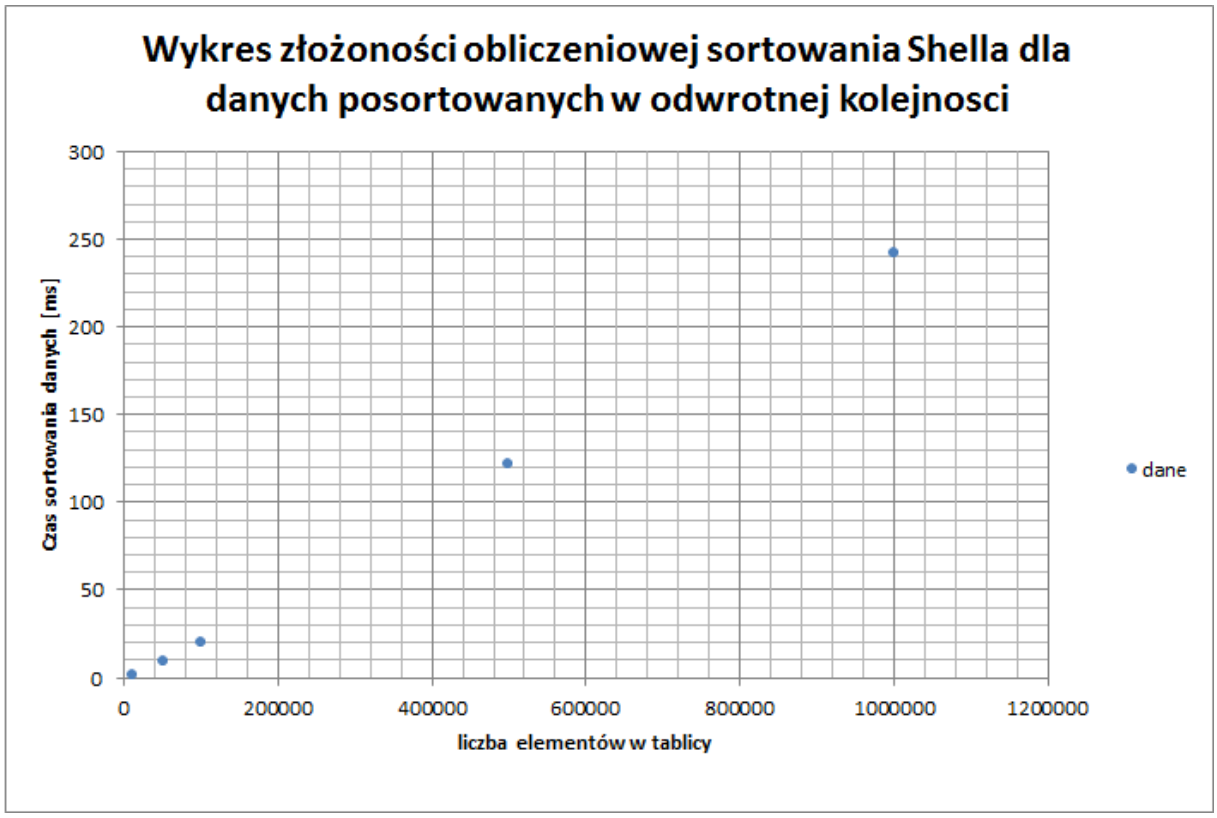
Algorytm sortowania Shella nie wykorzystuje rekurencji, dlatego też jego złożoność pamięciowa jest stała. Zasada działania polega na wstawianiu elementów odległych od siebie określoną długość. W czasie sortowania ta odległość jest systematycznie zmniejszana do zera. Złożoność obliczeniowa najgorszego przypadku wynosi n^2, a średniego $nlog(n)$.

Tabela 2. Badanie czasu sortowania danych

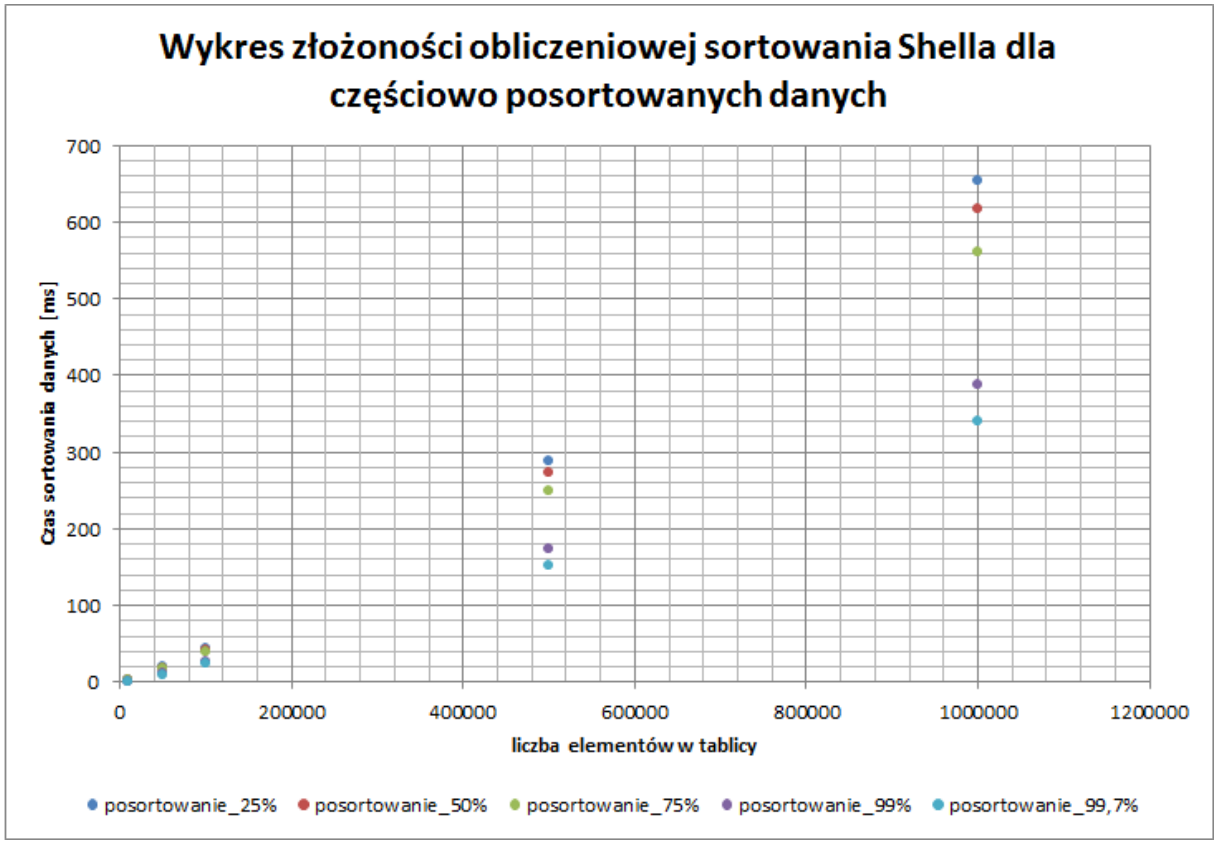
liczba elementów	czas sortowania [ms]						
	elementy losowe	Tablica posortowana	Posortowana część listy [%]				
			25	50	75	99	99,7
10000	3	1	2	2	2	1	1
50000	19	9	19	18	17	12	10
100000	45	20	44	42	39	27	24
500000	295	122	288	273	249	174	151
1000000	659	242	655	617	562	387	340



Rys.4. Wykres złożoności obliczeniowej sortowania Shella dla danych nieposortowanych



Rys.5. Wykres złożoności obliczeniowej sortowania Shella dla danych uporządkowanych w odwrotnej kolejności



Rys.6. Wykres złożoności obliczeniowej sortowania Shella dla danych częściowo posortowanych

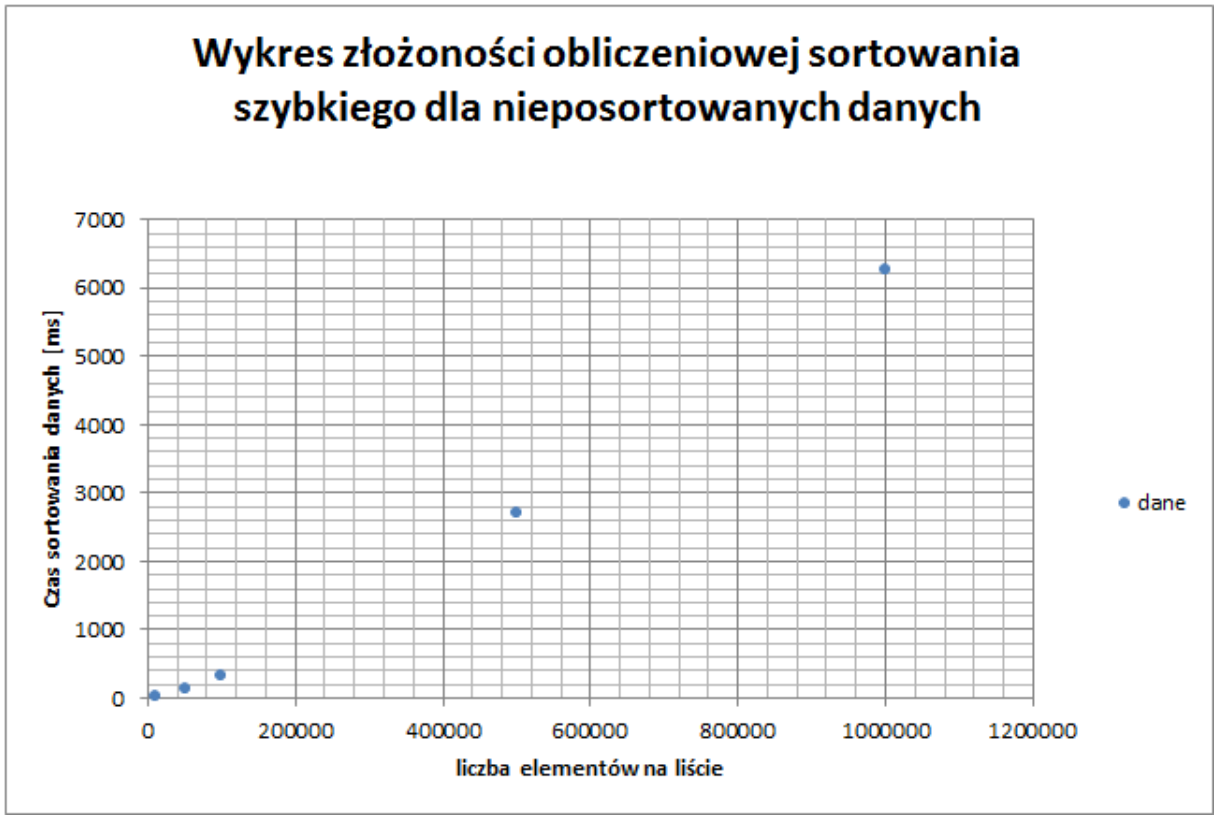
4. Badanie algorytmu sortowania szybkiego

Tak jak algorytm sortowania przez scalanie wykorzystuje metodę dziel i zwyciężaj, wywoływaną rekurencyjnie. Zasada działania polega na wyborze elementu osiowego (rozdzielającego) tzw. piwota, następnie ciąg danych dzielony jest na dwie części. Do pierwszej części wpisywane są elementy mniejsze od wartości piwota, a do drugiej elementy większe. Potem każda z tych części jest sortowana osobno. Rekurencja zakończy się po uzyskaniu jednoelementowego zestawu danych.

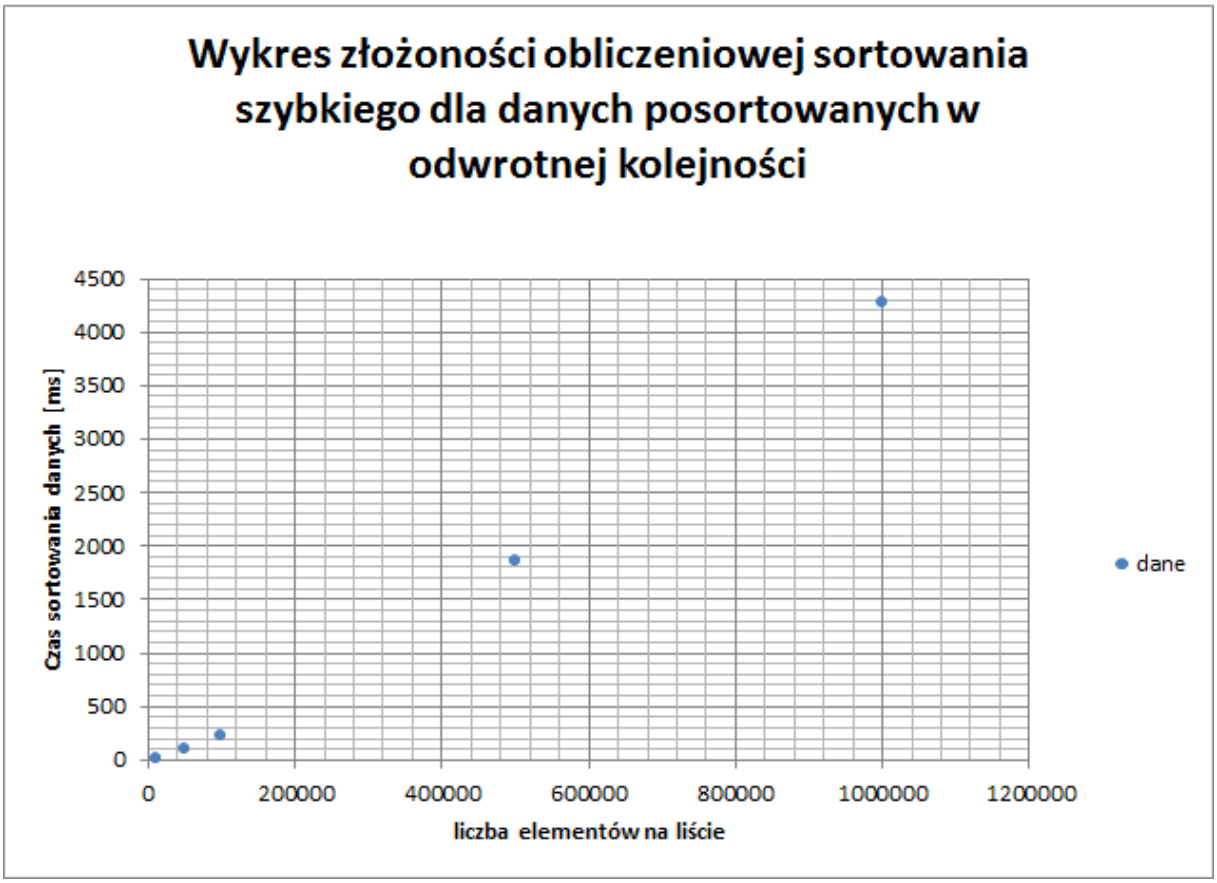
Złożoność obliczeniowa najgorszego przypadku wynosi n^2, a średniego $nlog(n)$.

Tabela 3. Badanie czasu sortowania danych

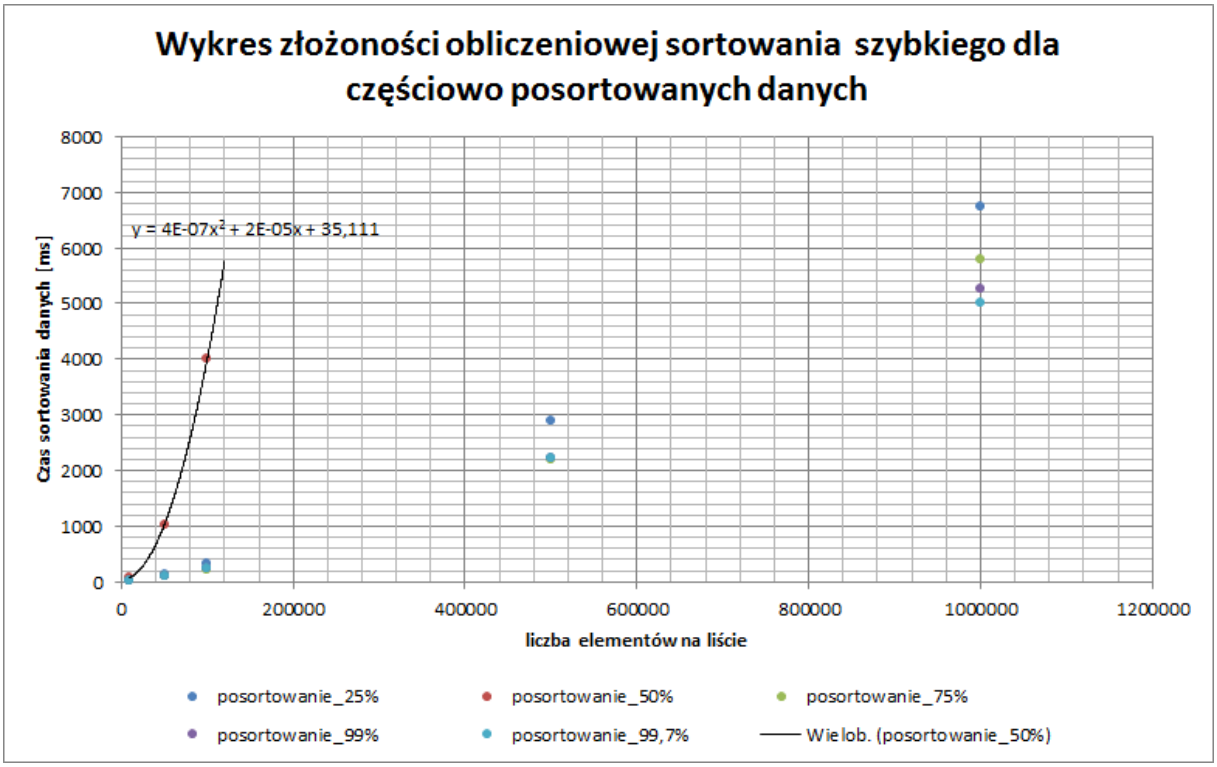
liczba elementów	czas sortowania [ms]						
	wszystkie elementy losowe	Lista posortowana w odwrotnej kolejności	Posortowana część listy [%]				
			25	50	75	99	99,7
10000	19	14	21	75	16	14	14
50000	129	95	134	1029	113	98	96
100000	318	229	331	4009	227	250	250
500000	2699	1854	2882	b. dużo	2207	2234	2226
1000000	6267	4276	6733	b.dużo	5799	5251	5027



Rys.7. Wykres złożoności obliczeniowej sortowania szybkiego dla danych nieposortowanych



Rys.8. Wykres złożoności obliczeniowej sortowania szybkiego dla danych uporządkowanych w odwrotnej kolejności



Rys.9. Wykres złożoności obliczeniowej sortowania szybkiego dla danych częściowo posortowanych

5. Wnioski

- Sortowanie przez scalanie

Na podstawie wykresów złożoności obliczeniowej sortowania przez scalanie można stwierdzić, że dla każdego przypadku złożoność obliczeniowa wynosi $nlog(n)$, co jest zgodne z założeniami. Największy rozrzut wyników pomiarów dla Rys.3. występuje przy milionie elementów.

- Sortowanie Shella

Z wykresów dla sortowania Shella wynika, że złożoność obliczeniowa wynosi w przybliżeniu $nlog(n)$, co jest zgodne z założeniami. Najdłuższy czas sortowania występuje dla listy zawierającej co najmniej 500 tyś elementów. Z Rys.6. można zauważyć, że dla zestawów danych o dużej liczbie elementów (>500 tyś.) największy wpływ na czas sortownia ma liczba już posortowanych elementów.

- Sortowanie szybkie

Na podstawie wykresów złożoności obliczeniowej sortowania szybkiego można stwierdzić, że najgorszy przypadek na złożoność obliczeniową n^2, a średni $nlog(n)$. Najgorsza złożoność obliczeniowa występuje dla zestawu danych, którego połowa elementów została posortowana. Przyczyną tego może być niewłaściwy dobór piwota. W takiej sytuacji piwot jest najmniejszą lub największą liczbą z zakresu danych. Skutkuje to nierównomiernym podziałem (jeden z podciągów jest dużo razy większy od drugiego) co wydłuża czas obliczeń.

6. Literatura

Strony internetowe:
https://pl.wikipedia.org/
http://www.bogotobogo.com/Algorithms/algorithms.php
http://www.algorytm.edu.pl