Politechnika Wrocławska

Projektowanie algorytmów i metody sztucznej inteligencji
Autor: Piotr Kuboń 252871
Kod grupy: E12-99c
Prowadzący: Mgr inż. Marta Emirsajłow
Data zajęć: 26.03.2021

1. Wprowadzenie

Celem projektu była analiza 3 różnych algorytmów sortowań zaimplementowanych w C++. Analizę wykonano pod względem ich złożoności obliczeniowej. Określa ona przybliżony czas wykonania algorytmu niezależnie od zastosowanego sprzętu.

Analizie poddano algorytm sortowania przez scalanie, sortowanie Shella oraz sortowanie szybkie.

Odbyła się ona poprzez pomiar czasu sortowania 100 tablic o 5 różnych rozmiarach, jak również rozpatrywano takie przypadki jak:

- Wszystkie elementy tablicy losowe
- Posortowane 25%, 50%, 75%, 95%, 99%, 99.7% początkowych elementów tablicy
- Wszystkie elementy posortowane w odwrotnej kolejności

2. Opis algorytmów sortujących

2.1. Sortowanie szybkie

Jest to algorytm stosujący strategię „dziel i zwyciężaj" opierający się na dzieleniu problemu na mniejsze części a następnie rozwiązywaniu każdego z nich oddzielnie. Niezbędne do realizacji algorytmu jest wyznaczenie elementu dzielącego, tzw. „pivota" który posłuży do podzielenia tablicy na część elementów mniejszych i część elementów większych od niego. Następnie przy użyciu rekurencji sortowanie wywoływane jest dla kolejnych zbiorów. Z najlepszym przypadkiem mamy do czynienia w momencie kiedy pivot jest medianą sortowanego fragmentu tablicy. Mamy wtedy do czynienia z złożonością obliczeniową na poziomie $nlogn$. Najmniej korzystny przypadek mamy wtedy gdy za każdym razem pivot jest elementem największym, lub najmniejszym. Złożoność obliczeniowa wzrasta wtedy do poziomu n^2 . W celu zmniejszenia prawdopodobieństwa wystąpienia pesymistycznego przypadku stosuje się m .in. algorytm wyboru pivota pt. „Median of three".

2.2. Sortowanie przez scalanie

Tak jak w przypadku powyżej mamy do czynienia z algorytmem wykorzystującym rekurencję do podziału tablicy, jednak w tym przypadku powoduje ono uzyskanie tablic jednoelementowych, które następnie scala się do tablicy początkowej, jednak na etapie pośrednim następuje porównanie i przestawienie elementów w celu ich poprawnego zapisu. W przypadku tego algorytmu mamy do czynienia ze złożonością obliczeniową na poziomie $nlogn$.

2.3. Sortowanie Shella

Jest to sortowanie bazujące bardzo mocno na sortowaniu przez wstawanie, jednak różni się od niego faktem posiadania parametru określającego odległość pomiędzy porównywanymi elementami. Odległość ta zmniejsza się wraz z ilością iteracji zbioru. Złożoność obliczeniowa algorytmu Shella wynosi $mN^{1+\frac{1}{m}}$ lub w najgorszym przypadku (dla $m > \log_2 N$) mN .

Sortowanie to wykonuje więcej operacji porównania niż sortowanie szybkie.

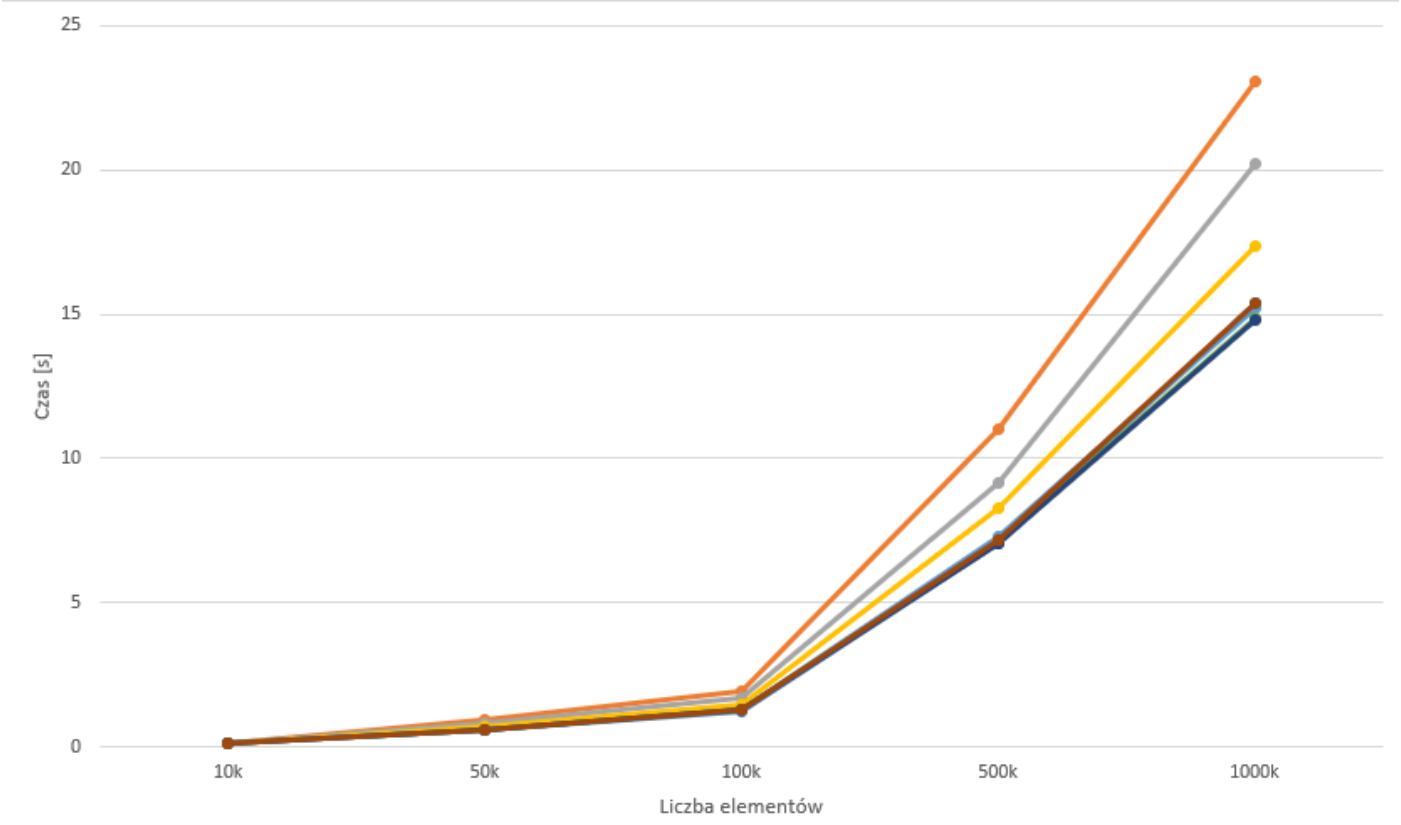
3. Prezentacja wyników

Przedstawione wykresy oraz tabele dotyczą uśrednionego czasu wykonania dla 100 tablicy danego rozmiaru.

3.1. Sortowanie przez scalanie.

Czas został podany w sekundach.

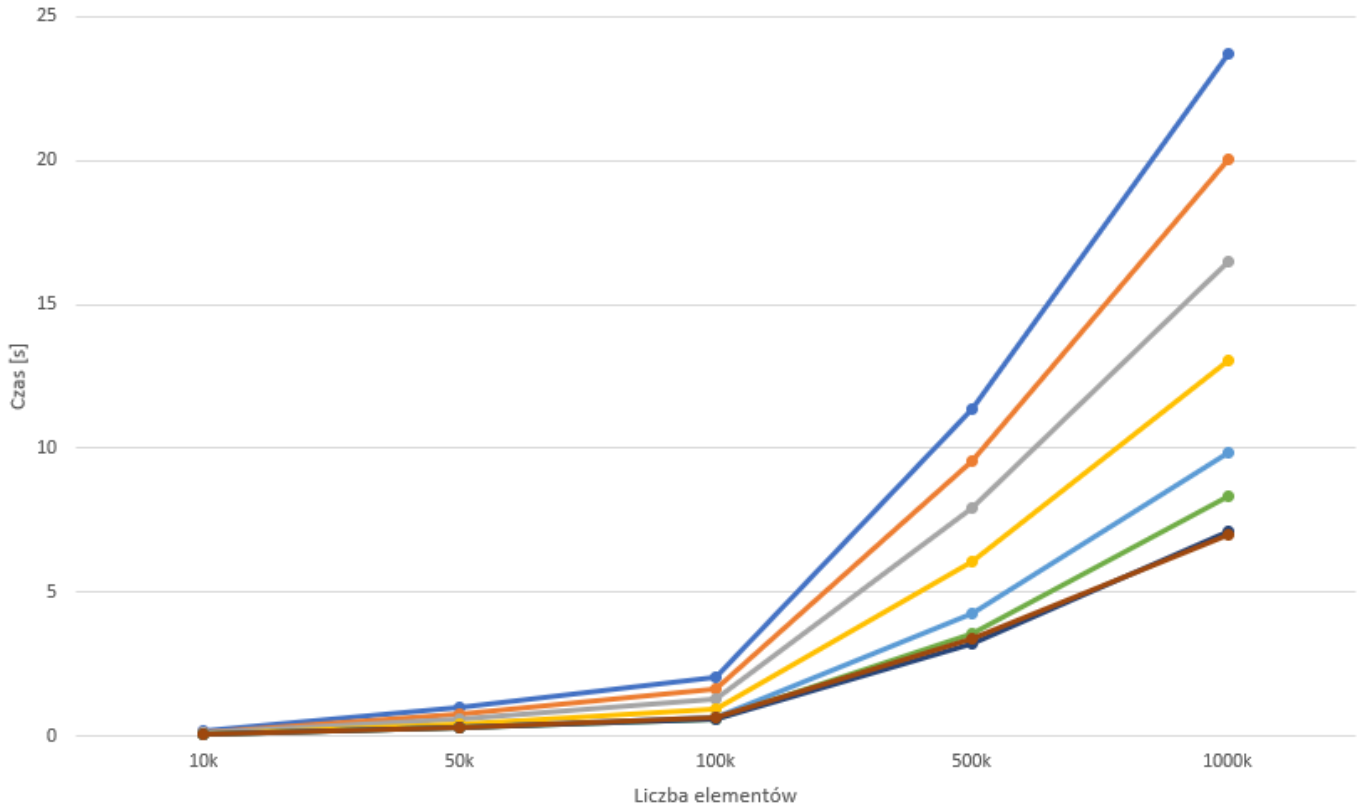
Liczba elementów	Poziom posortowania							
	0%	25%	50%	75%	95%	99%	99,7%	Odwrotna kolejność
10k	0,1065	0,161	0,1414	0,1294	0,1131	0,1070	0,1096	0,1084
50k	0,5989	0,9278	0,811	0,705	0,621	0,6047	0,6048	0,6121
100k	1,26046	1,95	1,714	1,48	1,306	1,2716	1,2772	1,3102
500k	7,0775	11,038	9,162	8,26	7,27	7,1039	7,0632	7,1879
1000k	15,3629	23,06	20,188	17,33	15,23	14,872	14,822	15,4081



3.2. Sortowanie szybkie

Czas został podany w sekundach.

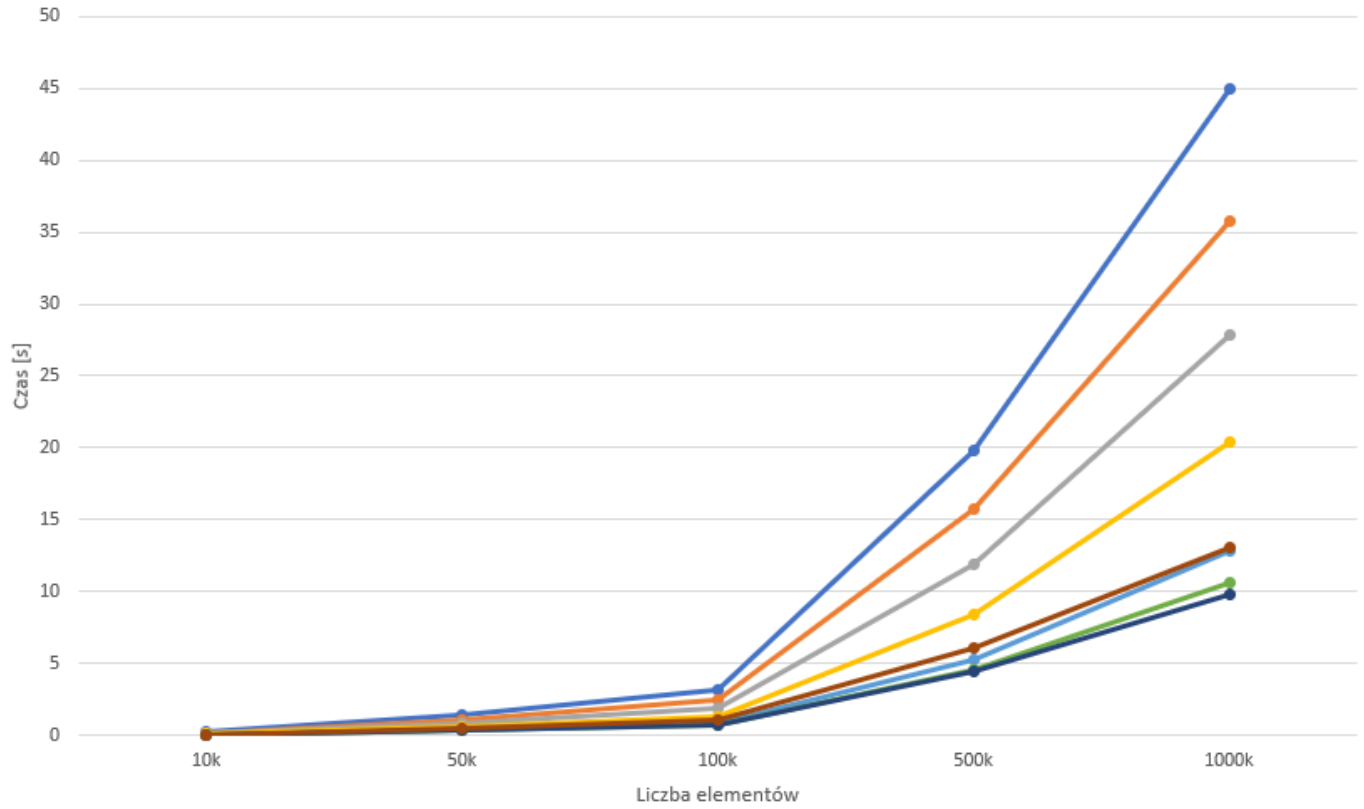
Liczba elementów	Poziom posortowania							
	0%	25%	50%	75%	95%	99%	99,7%	Odwrotna kolejność
10k	0,176	0,1383	0,1123	0,079	0,0584	0,05	0,0514	0,055
50k	0,9778	0,7845	0,6144	0,4437	0,337	0,296	0,294	0,322
100k	2,0318	1,67	1,31	0,9472	0,667	0,615	0,624	0,653
500k	11,36	9,538	7,9066	6,051	4,2836	3,545	3,224	3,38
1000k	23,69	20,024	16,51	13,047	9,86	8,35	7,116	7,0218



3.3. Sortowanie Shella

Czas został podany w sekundach.

Liczba elementów	Poziom posortowania							
	0%	25%	50%	75%	95%	99%	99,7%	Odwrotna kolejność
10k	0,2269	0,177	0,1369	0,0978	0,0697	0,0636	0,0626	0,089
50k	1,421	1,127	0,8572	0,5715	0,396	0,371	0,36	0,516
100k	3,185	2,527	1,8513	1,265	0,8496	0,788	0,7804	1,105
500k	19,85	15,689	11,944	8,399	5,324	4,583	4,422	6,104
1000k	44,933	35,78	27,8994	20,412	12,81	10,589	9,76	13,047



3.4. Wnioski z zaprezentowanych wyników:

- Algorytmy zachowują swoją złożoność obliczeniową
- Sortowanie szybkie jest najszybszym algorytmem sortowania dla danych częściowo uporządkowanych i o znacznych rozmiarach

4. Wnioski ogólne

- Szybkość wykonywania wszystkich algorytmów sortowania zależy od wielkości zbiorów oraz ich pierwotnego uporządkowania jak również w szczególności od ich ilości.
- Należy dobierać rodzaj sortowania do określonego rodzaju uporządkowania danych aby zminimalizować czas jaki zajmie algorytmowi ich sortowanie.
- Algorytmy o zbliżonej złożoności obliczeniowej realizują cel w zbliżonym czasie.

5. **Literatura**

- https://en.wikipedia.org/wiki/Shellsort
- https://en.wikipedia.org/wiki/Quicksort
- https://en.wikipedia.org/wiki/Merge_sort
- https://www.youtube.com/watch?v=yzkzQ7oZwIE
- https://www.youtube.com/watch?v=82XxdhRCMbI
- https://www.youtube.com/watch?v=3yUSDJVDk4E&t=820s
- https://www.youtube.com/watch?v=SHcPqUe2GZM
- https://stackoverflow.com