Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Projekt 1
Algorytmy sterowania

Krzysztof Górski, 245079
prowadzący Mgr Marta Emirsajłow

grupa piątek, 13:15-15:00

1 Wstęp

1.1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się, implementacją oraz eksperymentalne testowanie algorytmów sortowania, polegające na badaniu złożoności czasowej algorytmu podczas sortowania 100 tablic elementów typu całkowitoliczbowego o różnych rozmiarach oraz róznych stopniach pososrtowania, jak również rozpatrzenie najgorszych przypadków dla poszczególnych algorytmów.

2 Algorytmy

2.1 Sortowanie przez scalanie - Merge Sort

Rekurencyjny algorytm sortowania danych, stosujący metodę dziel i zwyciężaj. Tablica w każdym kroku zostaje podzielona na dwie części, aż do powstania tablic jednoelementowych. Po wystąpieniu przypadku bazowego, algorytm porównuje obie tablice, a następnie scala je, układając ich elementy w odpowiedniej kolejności.
Algorytm posiada następującą złożoność obliczeniowa:

- **średni przypadek** - $O(nlogn)$ jest to równo głębokości kompletnego drzewa binarnego

- **najlepszy przypadek** -$O(nlogn)$**najgorszy przypadek**– O(nlog n) .

2.2 Sortowanie przez wstawianie - Insert Sort

jeden z najprostszych algorytmów sortowania, kolejne elementy wejściowe są ustawiane na odpowiednie miejsca docelowe. Jest efektywny dla niewielkiej liczby elementów.
Posiada następującą złożoność obliczeniowa:

- **średni przypadek** - $O(n^2)$ porownań i podstawień
- **najlepszy przypadek** - $O(n)$ porównan, $O(1)$ podstawień.
- **najgorszy przypadek** - $O(n^2)$ porownań i podstawień

2.3 Sortowanie Szybkie - Quicksort

Rekurencyjny algorytm sortowania danych, stosujący metodę dziel i zwyciężaj. Z tablicy wybiera się element rozdzielający, po czym tablica jest dzielona na dwa fragmenty: do początkowego przenoszone są wszystkie elementy nie większe od rozdzielającego, do końcowego wszystkie większe. Potem sortuje się osobno początkową i końcową część tablicy[1]. Rekursja kończy się, gdy kolejny fragment uzyskany z podziału zawiera pojedynczy element.
Algorytm posiada następującą złożoność obliczeniowa:

- **średni przypadek** - $O(nlogn)$
- **najlepszy przypadek** - $O(nlogn)$
- **najgorszy przypadek** - $O(n^2)$ przypadek gdy tablica jest posortowana odwrotnie.

2.4 Sortowanie Introspektywne - Intro Sort

Hybrydowy algorytm sortowania, który łaczy ze sobą algorytm sortowania szybkiego z algorytmem sortowania przez kopcowanie i/lub sortowania przez wstawiane. Algorytm ma na celu wyelimininowanie najgorszego przypadku dla sortowania szybkiego.Algorytm działa jak sortowanie szybkie do osiągnięcia maksymalnej dozwolonej głębokości wywołań rekurencyjnych. Następnie wywoływana jest procedura Intro Sort/CheapSort
Algorytm posiada następującą złożoność obliczeniowa:

- **średni przypadek** - $O(nlogn)$ taki sam jak dla sortowania szybkiego
- **najlepszy przypadek** - $O(nlogn)$
- **najgorszy przypadek** - $O(nlogn)$ algorytm eliminuje problem tablicy posortowanej odwrotnie

3 Pomiary

ekpseryment polegał na porównaniu algorytmów, przeprowadzono pomiary czasu sortowania 100 tablic o rozmiarach:
10.000, 50.000, 100.000, 500.000 i 1.000.000
, oraz różnych stopniach wstępnego posortowania:
$0\%, 25\%, 50\%, 75\%, 95\%, 99\%, 99.7\%$ oraz tablicy posortowanej odwrotnie.

W celu porównania algorytmów odmierzono czas sortowania stu tablic przez dany algorytm. W tym celu zastosowano bibliotekę *chrono*. Następnie obliczono średni czas sorotwania jednej tablicy.

Pomiary czasu [s] stu tablic dla metody Merge Sort

wielkość tablicy \ stopień posortowania	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,65695	0,51915	0,50005	0,48166	0,47527	0,47923	0,47659	0,54431
50000	3,29246	2,63967	2,54687	2,60030	2,54612	2,41413	2,38703	2,43994
100000	6,00097	5,39797	5,19901	5,07042	4,85043	5,08279	5,04218	5,02272
500000	30,8280	27,6469	26,0768	25,4697	25,1381	25,2130	25,1729	25,4749
1000000	60,4233	55,4890	52,7890	51,6050	51,0185	50,8227	50,9158	51,1695

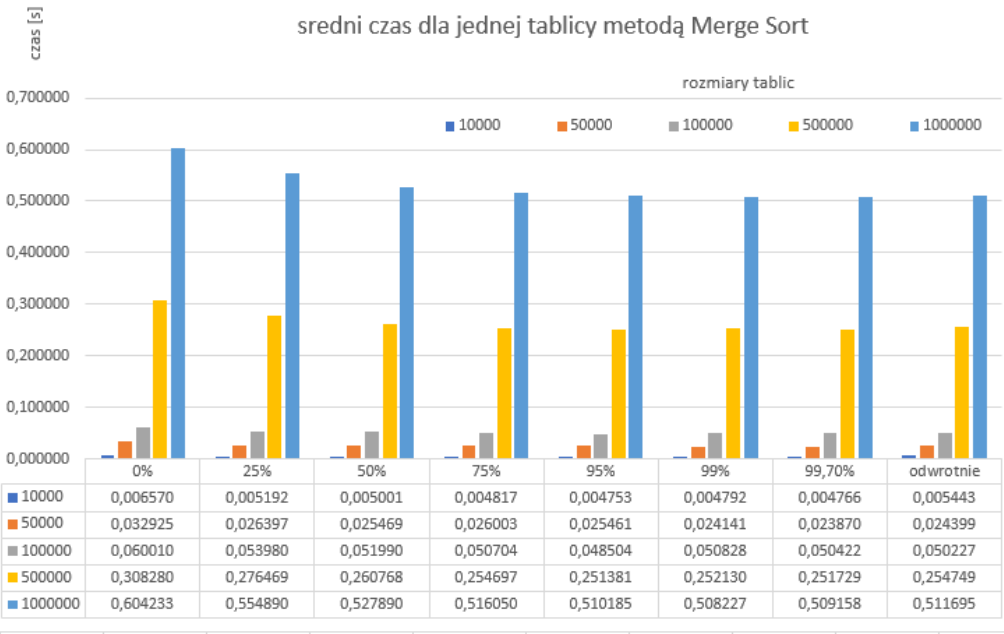
Pomiary czasu [s] stu tablic dla metody Intro Sort

wielkość tablicy \ stopień posortowania	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,12858	0,10682	0,08118	0,06195	0,05321	0,04939	0,05844	0,05562
50000	0,73602	0,56179	0,45266	0,36165	0,33275	0,31626	0,31804	0,30861
100000	1,44195	1,29131	0,92786	0,78790	0,77568	0,67825	0,68032	0,68819
500000	7,27994	5,93044	4,83580	3,96810	3,51025	3,51533	3,49754	3,52167
1000000	14,26810	12,23660	9,39114	7,80902	7,03330	7,22209	7,19033	8,06028

Quick Sort dla 100 tablic

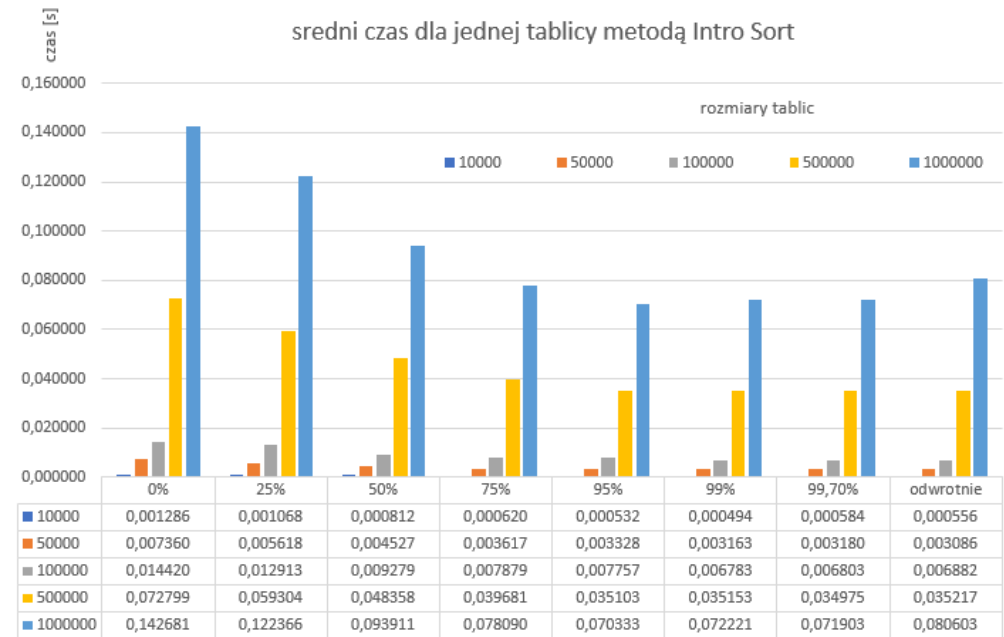
wielkość tablicy \ stopień posortowania	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,20201	0,10381	0,08946	0,06166	0,06564	0,05321	0,04913	0,05924
50000	0,70892	0,61011	0,50788	0,37724	0,33090	0,32712	0,32800	0,33483
100000	1,41105	1,19416	0,96435	0,77787	0,71559	0,72474	0,74936	0,71936
500000	7,50044	5,74398	4,70602	4,03586	3,55610	3,61913	3,64831	3,60290
1000000	14,35450	11,93800	10,97870	8,05495	7,34773	7,42365	7,26010	7,49303

Rysunek 1: Wyniki pomiaru czasu dla sortowań 100 tablic



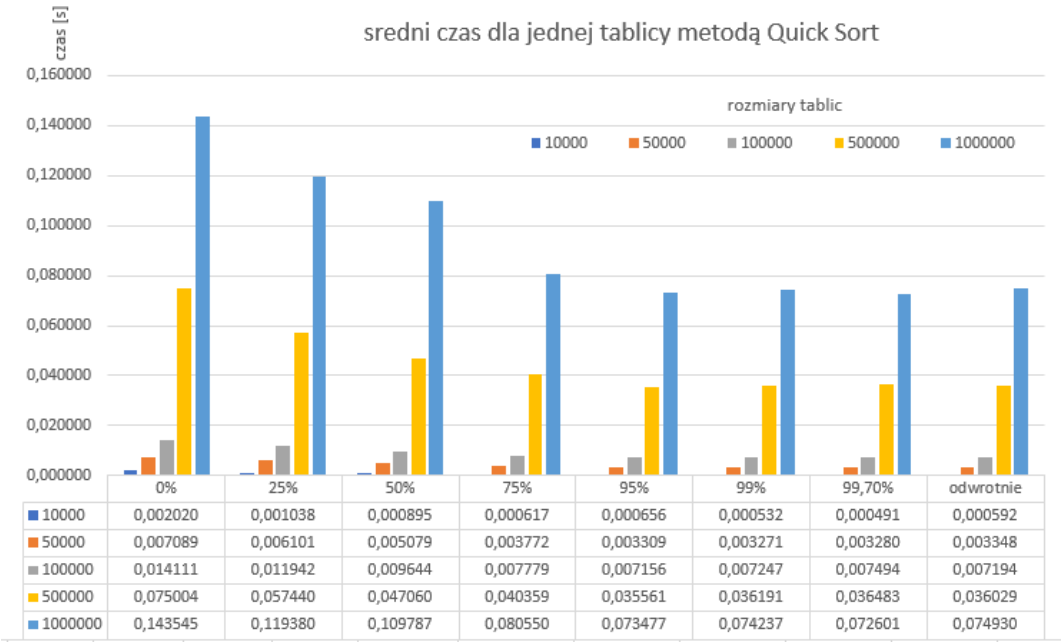
	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,006570	0,005192	0,005001	0,004817	0,004753	0,004792	0,004766	0,005443
50000	0,032925	0,026397	0,025469	0,026003	0,025461	0,024141	0,023870	0,024399
100000	0,060010	0,053980	0,051990	0,050704	0,048504	0,050828	0,050422	0,050227
500000	0,308280	0,276469	0,260768	0,254697	0,251381	0,252130	0,251729	0,254749
1000000	0,604233	0,554890	0,527890	0,516050	0,510185	0,508227	0,509158	0,511695

Rysunek 2: Sredni czas sortowania jednej tablicy dla algorytmu Merge Sort



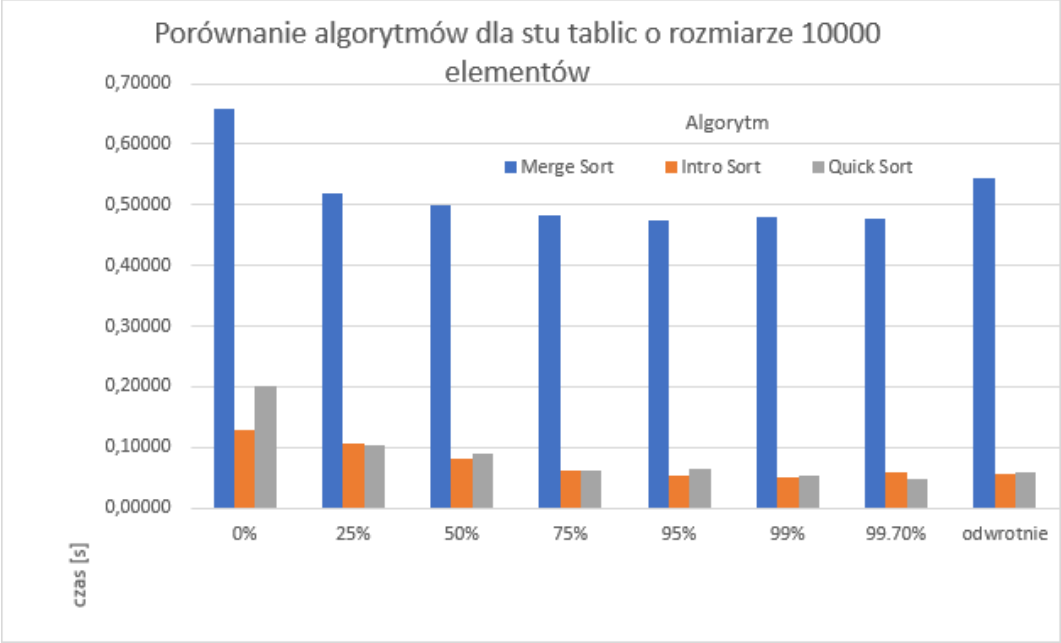
	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,001286	0,001068	0,000812	0,000620	0,000532	0,000494	0,000584	0,000556
50000	0,007360	0,005618	0,004527	0,003617	0,003328	0,003163	0,003180	0,003086
100000	0,014420	0,012913	0,009279	0,007879	0,007757	0,006783	0,006803	0,006882
500000	0,072799	0,059304	0,048358	0,039681	0,035103	0,035153	0,034975	0,035217
1000000	0,142681	0,122366	0,093911	0,078090	0,070333	0,072221	0,071903	0,080603

Rysunek 3: Sredni czas sortowania jednej tablicy dla algorytmu Intro Sort

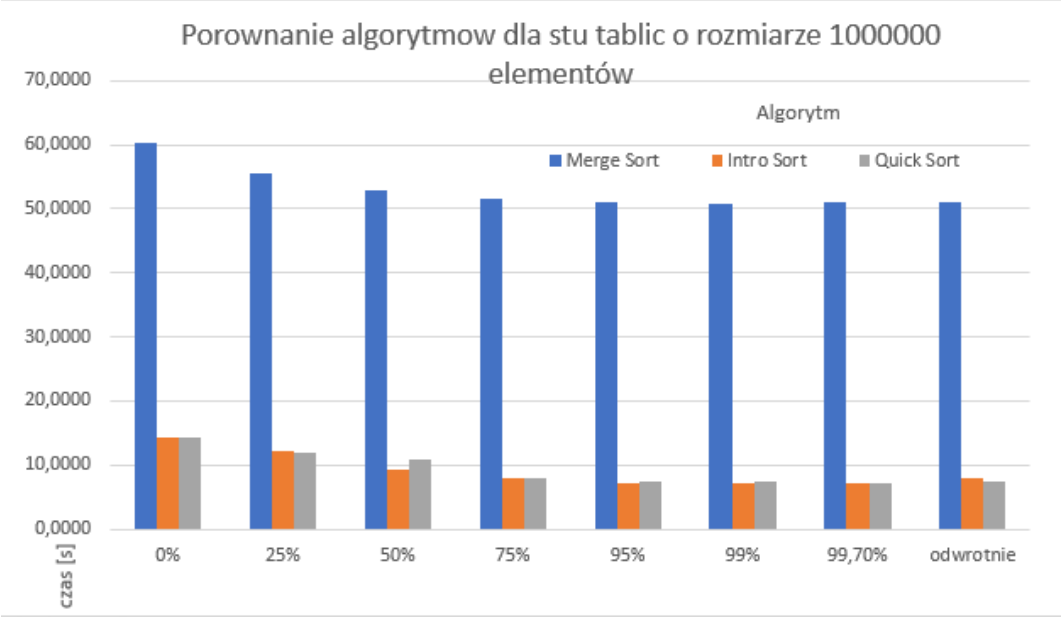


	0%	25%	50%	75%	95%	99%	99,70%	odwrotnie
10000	0,002020	0,001038	0,000895	0,000617	0,000656	0,000532	0,000491	0,000592
50000	0,007089	0,006101	0,005079	0,003772	0,003309	0,003271	0,003280	0,003348
100000	0,014111	0,011942	0,009644	0,007779	0,007156	0,007247	0,007494	0,007194
500000	0,075004	0,057440	0,047060	0,040359	0,035561	0,036191	0,036483	0,036029
1000000	0,143545	0,119380	0,109787	0,080550	0,073477	0,074237	0,072601	0,074930

Rysunek 4: Sredni czas sortowania jednej tablicy dla algorytmu Quick Sort



Rysunek 5: Porownanie czasów sortowania stu tablic o rozmiarze 10000 elementów dla róznych algorytmów



Rysunek 6: Porownanie czasów sortowania stu tablic o rozmiarze 1000000 elementów dla róznych algorytmów

4 Wnioski

Zgodnie z przewidywaniami najszybszym algorytmem jest algorytm IntroSort, Quicksort ma bardzo zbliżone, niewiele dłuższe czas sortowania. Najwolniejszym algorytmem jest algorytm MergeSort. Prawdopodonie wynika to z faktu, że w za implementacji użyto dynamicznie alokowanej tablicy do połączenia rekurencyjnych podtablic. Algorytm Intro Sort najlepiej sprawdza się przy tablicach mniejszych, w przypadku tablicy o rozmiarze miliona elementów wystąpiła anomalia. Zgodnie z przwidywaniami czas powinien być krótszy niż dla Qicksorta. Odstępstwo jest spowodowane prawdopodnie zaimplementowaniem w IntroSort jedynie sortowania przez wstawianie dla głębokich rekurencji. Zgodnie z przwidywaniami czas powinien być krótszy niż dla Qicksorta.