

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 różnych stopniach posortowania, 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 obliczeniową:

- **średni przypadek** - $O(n \log n)$ jest to równo głębokości kompletnego drzewa binarnego
- **najlepszy przypadek** - $O(n \log n)$ **najgorszy przypadek** - $O(n \log n)$.

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 obliczeniową:

- **średni przypadek** - $O(n^2)$ porównań i podstawień
- **najlepszy przypadek** - $O(n)$ porównań, $O(1)$ podstawień.
- **najgorszy przypadek** - $O(n^2)$ porównań 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 obliczeniową:

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

2.4 Sortowanie Introspektywne - Intro Sort

Hybrydowy algorytm sortowania, który łączy ze sobą algorytm sortowania szybkiego z algorytmem sortowania przez kopcowanie i/lub sortowania przez wstawianie. Algorytm ma na celu wyeliminowanie 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 obliczeniową:

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

3 Pomiary

eksperyment polegał na porównaniu algorytmów, przeprowadzono pomiary czasu sortowania tablic o rozmiarach: 10.000, 50.000, 100.000, 500.000 i 1.000.000

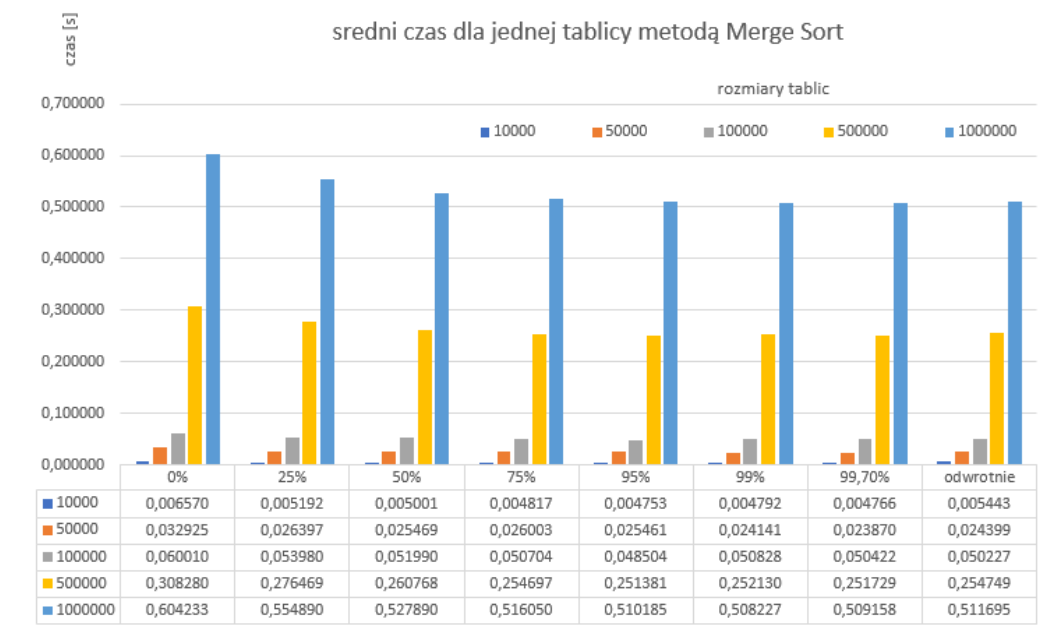
, oraz różnych stopniach wstępnego posortowania:

0%, 25%, 50%, 75%, 95%, 99%, 99.7% oraz tablicy posortowanej odwrotnie.

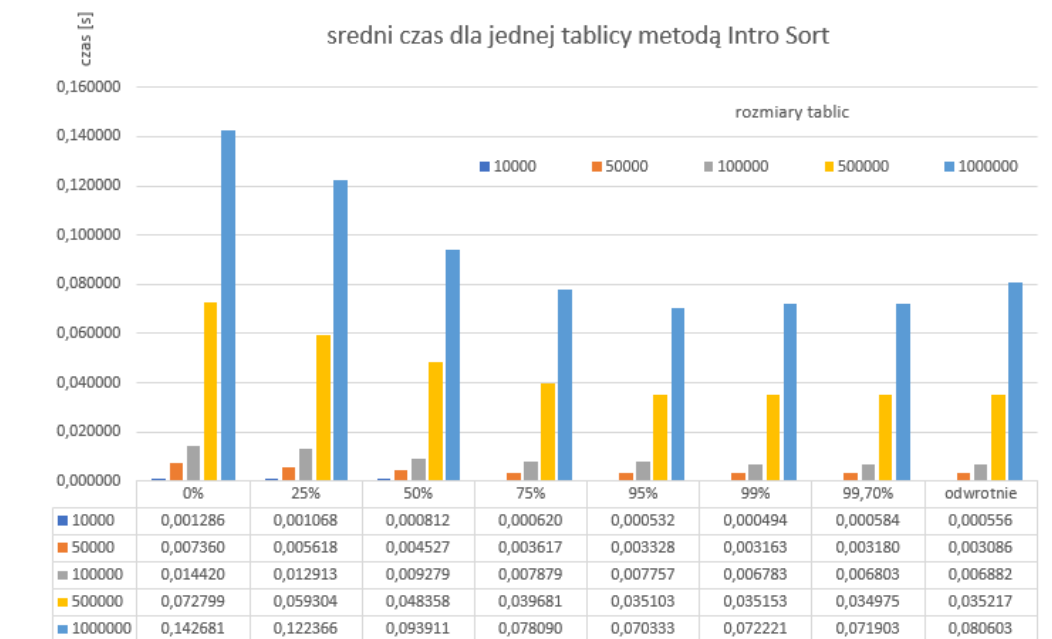
W celu porównania algorytmów odmierzonego czas sortowania jednej tablicy przez dany algorytm. W tym celu zastosowano bibliotekę *chrono*. Dla danego algorytmu wykonano 100 pomiarów dla każdej kombinacji rozmiaru tablicy oraz stopnia posortowania. W tabelach umieszczono średni wynik tych pomiarów.

| Pomiary czasu [s] stu tablic dla metody Merge Sort | | | | | | | | |
|--|--|----------|----------|----------|---------|---------|---------|---------|
| wielkość tablicy | stopień posortowania | | | | | | | |
| | | 0% | 25% | 50% | 75% | 95% | 99% | 99,70% |
| | 10000 | 0,65695 | 0,51915 | 0,50005 | 0,48166 | 0,47527 | 0,47923 | 0,47659 |
| | 50000 | 3,29246 | 2,63967 | 2,54687 | 2,60030 | 2,54612 | 2,41413 | 2,38703 |
| | 100000 | 6,00097 | 5,39797 | 5,19901 | 5,07042 | 4,85043 | 5,08279 | 5,04218 |
| | 500000 | 30,8280 | 27,6469 | 26,0768 | 25,4697 | 25,1381 | 25,2130 | 25,1729 |
| | 1000000 | 60,4233 | 55,4890 | 52,7890 | 51,6050 | 51,0185 | 50,8227 | 50,9158 |
| wielkość tablicy | Pomiary czasu [s] stu tablic dla metody Intro Sort | | | | | | | |
| | stopień posortowania | | | | | | | |
| | | 0% | 25% | 50% | 75% | 95% | 99% | 99,70% |
| | 10000 | 0,12858 | 0,10682 | 0,08118 | 0,06195 | 0,05321 | 0,04939 | 0,05844 |
| | 50000 | 0,73602 | 0,56179 | 0,45266 | 0,36165 | 0,33275 | 0,31626 | 0,31804 |
| | 100000 | 1,44195 | 1,29131 | 0,92786 | 0,78790 | 0,77568 | 0,67825 | 0,68032 |
| | 500000 | 7,27994 | 5,93044 | 4,83580 | 3,96810 | 3,51025 | 3,51533 | 3,49754 |
| | 1000000 | 14,26810 | 12,23660 | 9,39114 | 7,80902 | 7,03330 | 7,22209 | 7,19033 |
| wielkość tablicy | Quick Sort dla 100 tablic | | | | | | | |
| | stopień posortowania | | | | | | | |
| | | 0% | 25% | 50% | 75% | 95% | 99% | 99,70% |
| | 10000 | 0,20201 | 0,10381 | 0,08946 | 0,06166 | 0,06564 | 0,05321 | 0,04913 |
| | 50000 | 0,70892 | 0,61011 | 0,50788 | 0,37724 | 0,33090 | 0,32712 | 0,32800 |
| | 100000 | 1,41105 | 1,19416 | 0,96435 | 0,77787 | 0,71559 | 0,72474 | 0,74936 |
| | 500000 | 7,50044 | 5,74398 | 4,70602 | 4,03586 | 3,55610 | 3,61913 | 3,64831 |
| | 1000000 | 14,35450 | 11,93800 | 10,97870 | 8,05495 | 7,34773 | 7,42365 | 7,26010 |

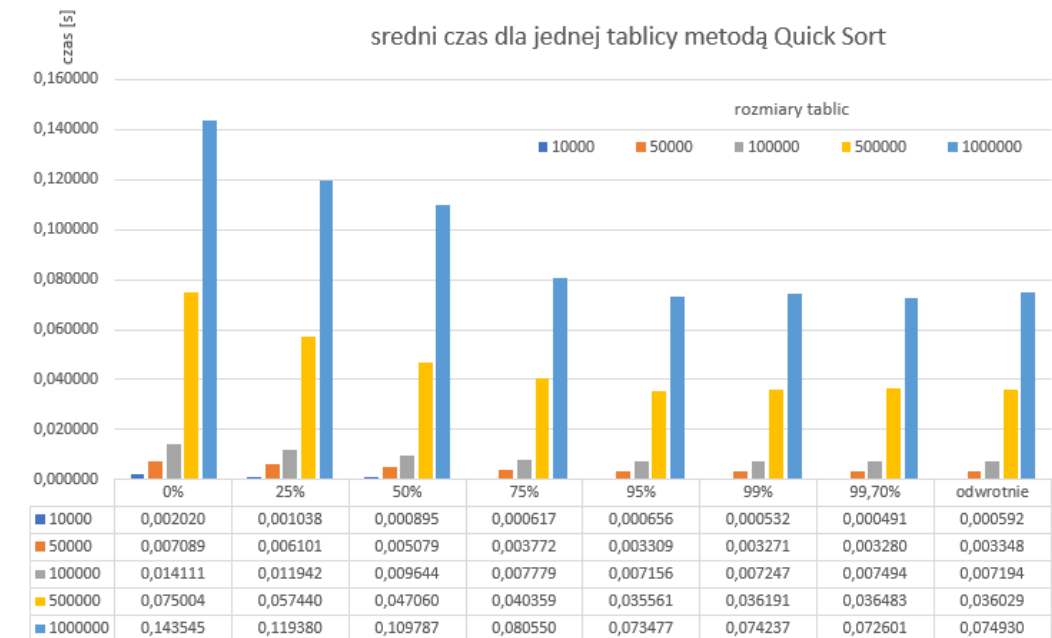
Rysunek 1: Wyniki pomiaru czasu dla sortowań 100 tablic



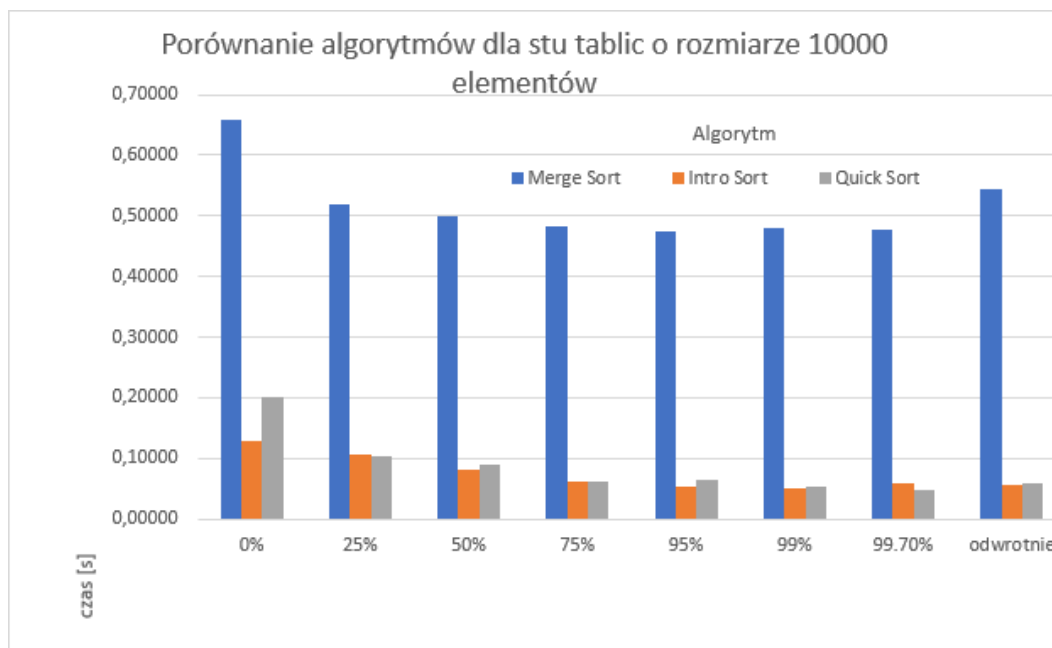
Rysunek 2: Sredni czas sortowania jednej tablicy dla algorytmu Merge Sort



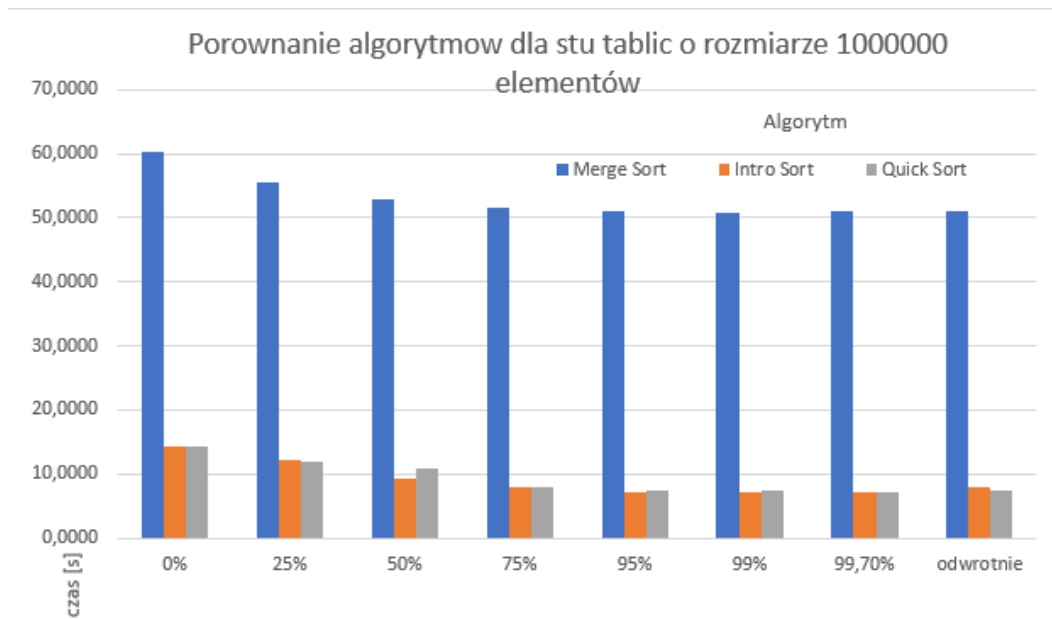
Rysunek 3: Sredni czas sortowania jednej tablicy dla algorytmu Intro Sort



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



Rysunek 5: Porównanie czasów sortowania stu tablic o rozmiarze 10000 elementów dla różnych algorytmów



Rysunek 6: Porównanie czasów sortowania stu tablic o rozmiarze 1000000 elementów dla różnych algorytmów

4 Wnioski

Zgodnie z przewidywaniami najszybszym algorytmem jest algorytm IntroSort, Quicksort ma bardzo zbliżone, niewiele dłuższe czas sortowania. Najwolniejszym algorytmem jest algorytm MergeSort. Prawdopodobnie 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 przewidywaniami czas powinien być krótszy niż dla Quicksorta. Odstępstwo jest spowodowane prawdopodobnie zaimplementowaniem w IntroSort jedynie sortowania przez wstawianie dla głębokich rekurencji. Zgodnie z przewidywaniami czas powinien być krótszy niż dla Quicksorta.