| Kierunek: | Nazwa zajęć: | Ocena: |
|---|---|---|
| **AIR** | **Projektowanie Algorytmów i Metody Sztucznej Inteligencji** | |

| Nr. projektu: | Tytuł projektu: |
|---|---|
| **1** | **Algorytmy sortowania (mergesort, quicksort, introsort)** |

| Termin: | Data oddania projektu: | Nr. grupy: |
|---|---|---|
| **Czwartek 17:05-18:45** | **26.03.2020 r.** | **-** |

| Osoby wykonujące projekt: | Podpisy: |
|---|---|
| **Maciej Salamoński** | |

| Sprawozdanie wykonał: | Maciej Salamoński |
|---|---|
| Data wykonania sprawozdania: | 25.03.2020 r. |
| Sprawozdanie sprawdził: | Dr inż. Krzysztof Halawa |

## 1. Cel projektu

Celem projektu było wykonanie eksperymentów z sortowaniem dla 100 tablic (elementy typu całkowitoliczbowego) o następujących rozmiarach: 10 000, 50 000, 100 000, 500 000, 1000 000 w następujących przypadkach:

- wszystkie elementy tablicy losowe,

- 25%, 50%, 75%, 95%, 99%, 99.7% początkowych elementów tablicy jest już posortowanych,

- wszystkie elementy tablicy już posortowane, ale w odwrotnej kolejności.

## 2. Wstęp teoretyczny

Sortowanie jest jednym z podstawowych problemów informatyki. Problem ten polega na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru np. wartości od najmniejszej do największej. Możemy sortować liczby, słowa itp. Algorytmy sortowania stosowane są, aby uporządkować dane, prezentować je w sposób czytelniejszy dla człowieka. Algorytmy te klasyfikowane są według złożoności obliczeniowej, złożoności pamięciowej, sposobu działania, stabilności. Algorytmy sortowania dzielimy na zewnętrzne i w miejscu. Zewnętrzne charakteryzują się tym, że sortowany zbiór danych przekracza wielkość dostępnej pamięci w odróżnieniu do sortowań w miejscu, gdzie nie jest potrzebna większa niż stała pamięć dodatkowa. Algorytmy sortujące możemy podzielić jeszcze na stabilne i niestabilne. Algorytmy sortujące, stabilne to takie, które dla elementów o tej samej wartości zachowują w tablicy końcowej kolejność tablicy wejściowej. Eksperymenty w ramach projektu zostały przeprowadzone bazując na algorytmach sortujących niestabilnych takich jak:

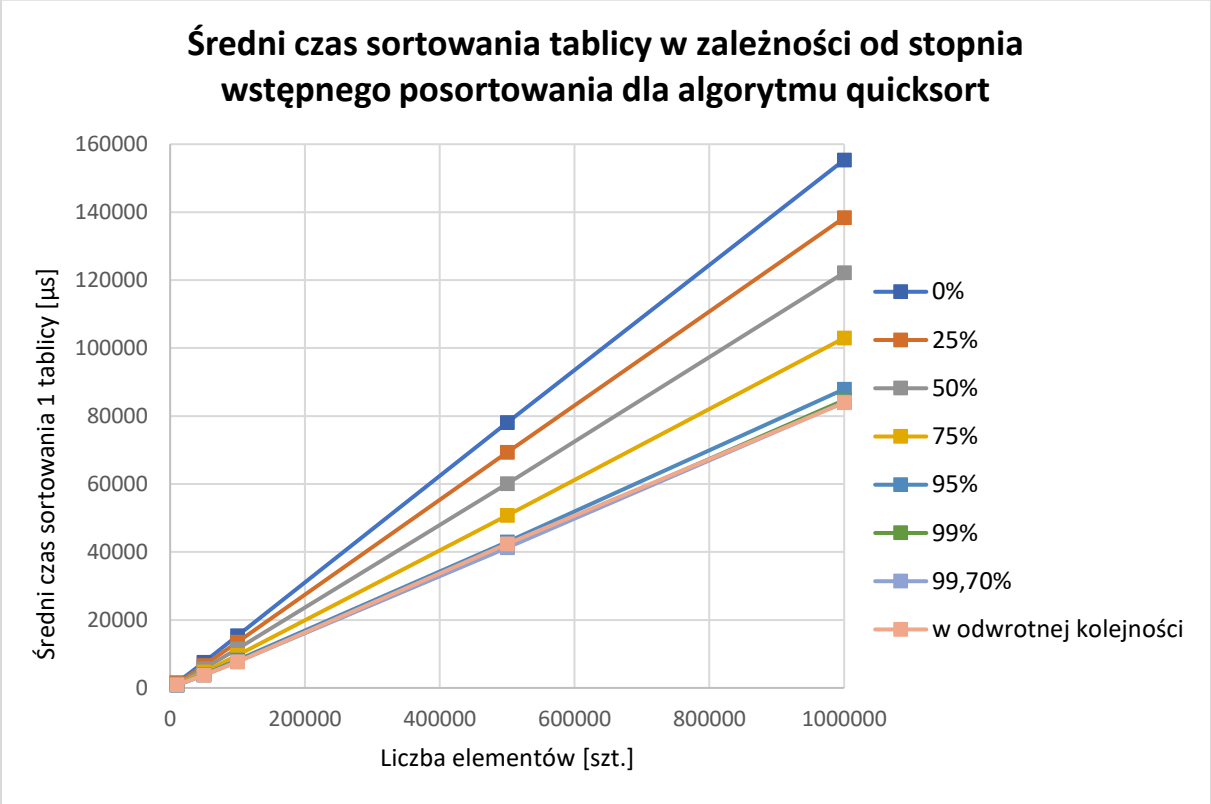
- sortowanie scalanie (mergesort)

- sortowanie szybkie (quicksort)

- sortowanie introspektywne (introsort)

## 3. Sortowanie szybkie (quicksort)

Wydajny algorytm sortowania. Sortowanie to należy do grupy algorytmów, które dzielą problem na podproblemy i wykorzystują rekurencję. Ma on liniowo logarytmiczną złożoność obliczeniową – $O(nlog(n))$, a w przypadku pesymistycznym kwadratową – $O(n^2)$.



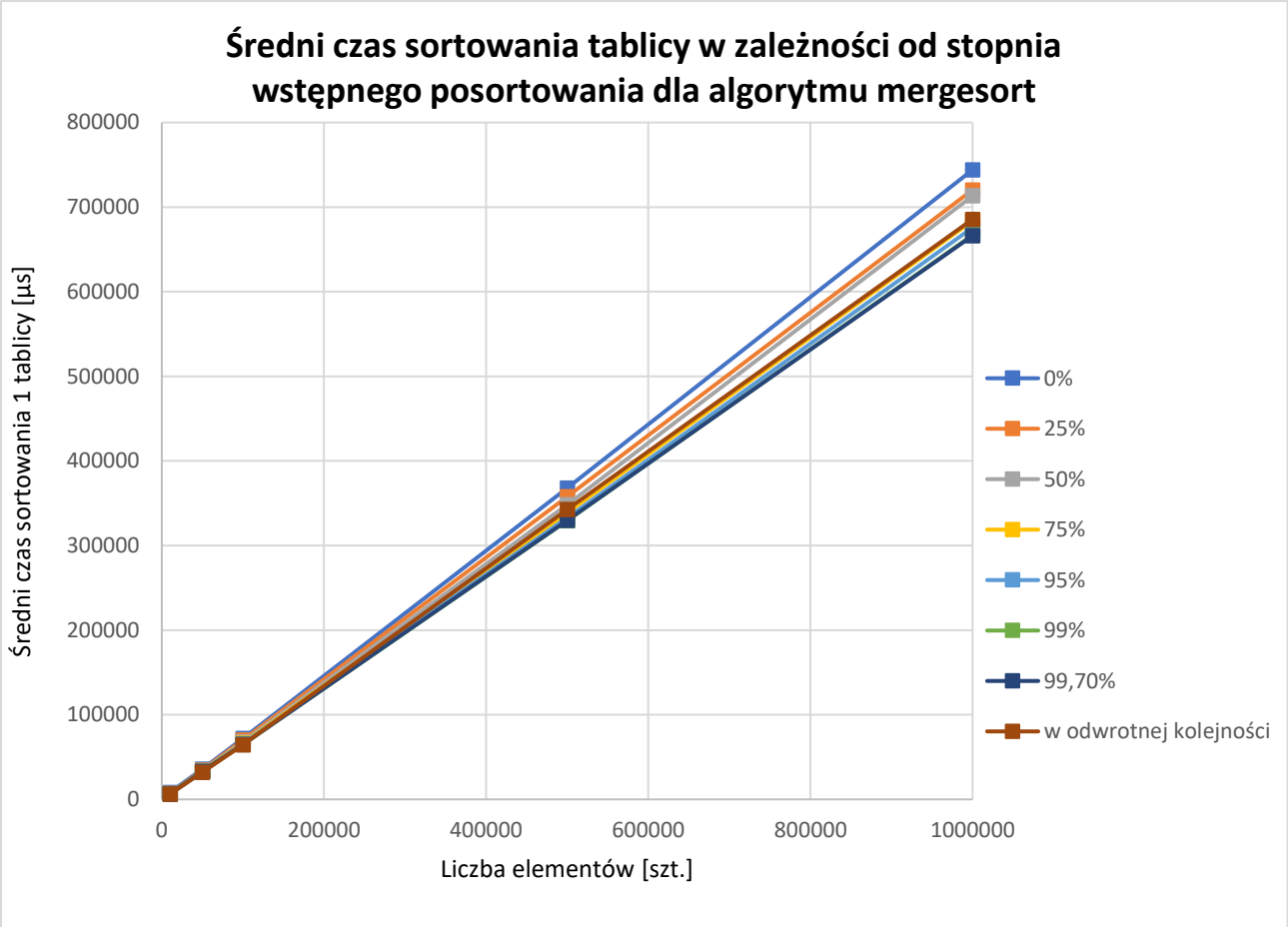

Wykres 1. Wykres prezentujący średni czas sortowania tablicy w zależności od stopnia wstępnego posortowania dla algorytmu quicksort.

| Tabela zbiorcza dla algorytmu sortowania szybkiego (quicksort) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Czas sortowania [µs] | | | | | | | |
| [%] / [szt.] | 0 | 25 | 50 | 75 | 95 | 99 | 99.7 | W odwrotnej kolejności |
| 10 000 | 1506 | 1273 | 1111 | 910 | 839 | 807 | 771 | 927 |
| 50 000 | 7571 | 6622 | 5611 | 4680 | 3970 | 3849 | 3784 | 3794 |
| 100 000 | 15373 | 13490 | 11472 | 9605 | 8156 | 7841 | 7753 | 7647 |
| 500 000 | 78076 | 69375 | 60146 | 50737 | 42913 | 41430 | 41218 | 42377 |
| 1000 000 | 155365 | 138396 | 122195 | 102960 | 87914 | 84688 | 84096 | 83921 |

Tabela 1. Tabela zbiorcza prezentująca czasy sortowania w zależności od liczby elementów oraz procentu wstępnego posortowania tablicy dla sortowania szybkiego (quicksort).

## 4. Sortowanie przez scalanie (mergesort)

Rekurencyjny algorytm sortowania, stosujący metodę dziel i zwyciężaj. Podobnie jak w algorytmie sortowania szybkiego, algorytm sortowania przez scalanie dzieli problem na mniejsze podproblemy i używa rekurencji. Ma on liniowo logarytmiczną złożoność obliczeniową – $O(nlog(n))$.



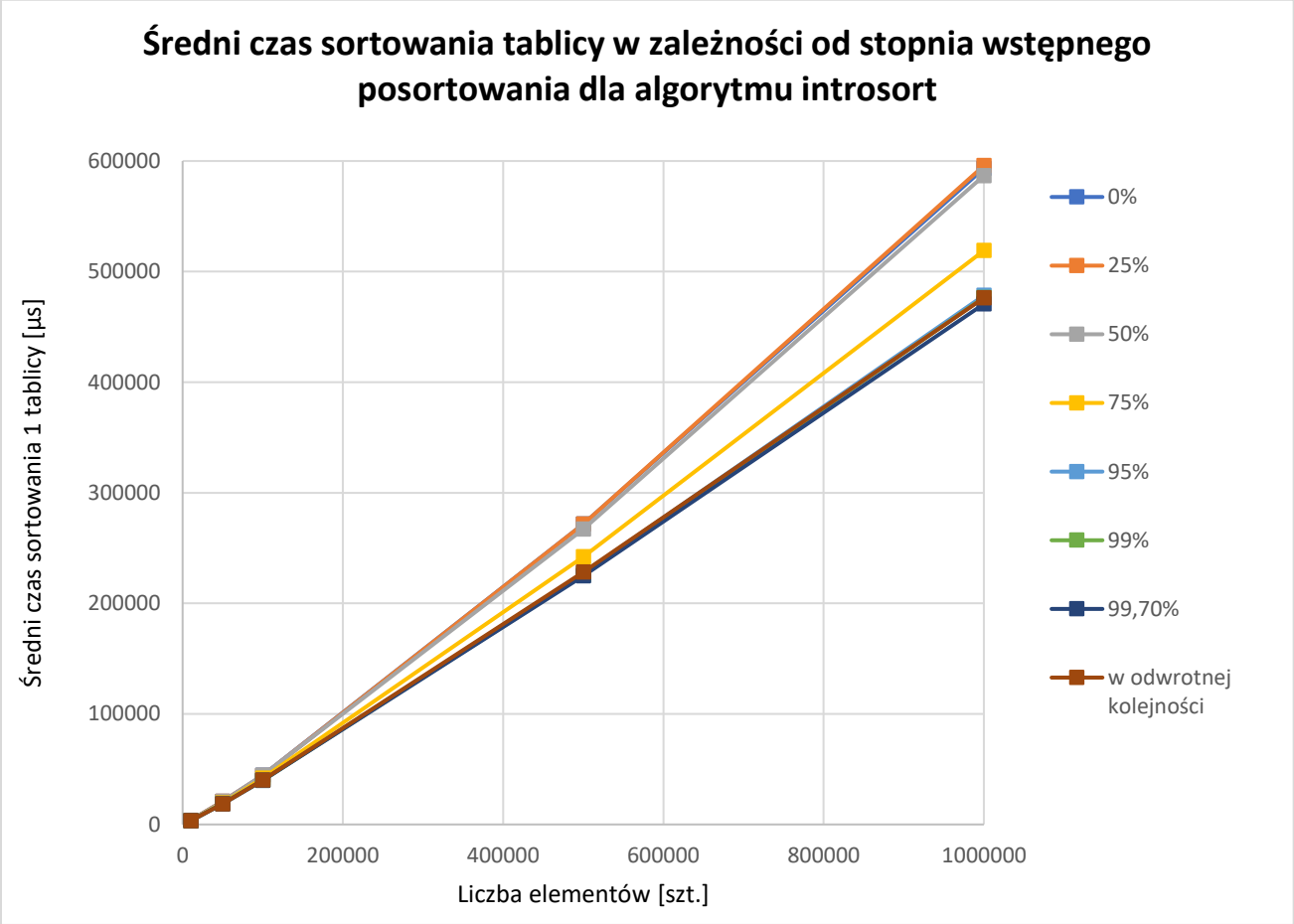

Wykres 2. Wykres prezentujący średni czas sortowania tablicy w zależności od stopnia wstępnego posortowania dla algorytmu mergesort.

| Tabela zbiorcza dla algorytmu sortowania przez scalanie (mergesort) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Czas sortowania [µs] | | | | | | | |
| [%] / [szt.] | 0 | 25 | 50 | 75 | 95 | 99 | 99.7 | W odwrotnej kolejności |
| 10 000 | 8316 | 6985 | 6577 | 6483 | 6298 | 6359 | 6357 | 6332 |
| 50 000 | 35705 | 34846 | 34118 | 33056 | 32433 | 33102 | 33049 | 32222 |
| 100 000 | 71949 | 70099 | 67983 | 66453 | 65606 | 65068 | 65108 | 64649 |
| 500 000 | 367911 | 357768 | 348304 | 338941 | 332905 | 329620 | 329903 | 342552 |
| 1000 000 | 744065 | 720339 | 713309 | 683905 | 675806 | 667003 | 666177 | 685650 |

Tabela 2. Tabela zbiorcza prezentująca czasy sortowania w zależności od liczby elementów oraz procentu wstępnego posortowania tablicy dla sortowania przez scalanie (mergesort).

## 5. Sortowanie introspektywne (introsort)

Sortowanie introspektywne jest odmianą sortowania hybrydowego. Bazuje on na algorytmie sortowania szybkiego. Wyeliminowany został w nim problem złożoności $O(n^2)$ występującej w najgorszym przypadku algorytmu quicksort. Uzyskano to poprzez dołączenia algorytmu sortowania przez kopcowanie (heapsort). Dzięki temu zabiegowi sortowanie introspektywne ma liniowo logarytmiczną złożoność obliczeniową – $O(nlog(n))$.




Wykres 3. Wykres prezentujący średni czas sortowania tablicy w zależności od stopnia wstępnego posortowania dla algorytmu introsort.

| Tabela zbiorcza dla algorytmu sortowania introspektywnego (introsort) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Czas sortowania [µs] | | | | | | | |
| [%] / [szt.] | 0 | 25 | 50 | 75 | 95 | 99 | 99.7 | W odwrotnej kolejności |
| 10 000 | 3726 | 3681 | 3592 | 3340 | 3260 | 3255 | 3254 | 3370 |
| 50 000 | 20934 | 20883 | 20564 | 19534 | 18900 | 18707 | 18664 | 18498 |
| 100 000 | 44553 | 44191 | 44430 | 41768 | 39889 | 39849 | 39905 | 40004 |
| 500 000 | 271575 | 271678 | 267182 | 241950 | 227314 | 225740 | 224757 | 228362 |
| 1000 000 | 593338 | 595790 | 586723 | 519000 | 478379 | 476603 | 470690 | 476146 |

Tabela 2. Tabela zbiorcza prezentująca czasy sortowania w zależności od liczby elementów oraz procentu wstępnego posortowania tablicy dla sortowania introspektywnego (introsort).

## 6. Wnioski

Po przeprowadzonych testach możemy dostrzec, że najszybciej z sortowaniem liczb poradził sobie algorytm sortowania szybkiego quicksort. Najgorzej poradził sobie z tym zadaniem algorytm sortowania przez scalanie (mergesort). Zastanawiać może fakt, że algorytm introspektywny, który jest ulepszeniem algorytmu sortowania szybkiego oraz jest on używany przez funkcje – sort() języka programowania c++ uplasował się za algorytmem quicksort. Pomimo zmniejszenia złożoności obliczeniowej do $O(nlog(n))$ dla każdego przypadku, nie uzyskano spodziewanego wyniku. Sortowanie introspektywne powinno wypaść najlepiej. Czynniki, które mogły mieć na to wpływ to np. implementacja algorytmu, która nie jest optymalna lub aplikacje uruchomione w tle podczas testów, które obciążyły procesor i spowodowały przekłamanie wyników.