# PROJEKTOWANIE ALGORYTMÓW SZTUCZNEJ INTELIGENCJI

# PROJEKT 2 – SORTOWANIE

08.04.2020r.

Sebastian Robak 248994

Prowadząca zajęcia: Mgr inż. Magda Skoczeń

Termin zajęć: Środa 18:55

# 1. Wstęp teoretyczny

Celem projektu o sortowaniu, było zapoznanie się z wybranymi algorytmami sortowania oraz ich złożoności obliczeniowej. Głównym zadaniem było przeprowadzenie testów napisanych algorytmów sortowania  w celu sprawdzenia ich złożoności obliczeniowej oraz porównania wyników z literaturą. Testy polegały na posortowaniu każdym z algorytmów 100 tablic o rozmiarach 10k, 50k, 100k, 500k oraz 1mln elementów, przy zmianie części początkowych elementów, która jest posortowana od 0 do 99.7% oraz przypadku w pełni posortowanej tablicy w odwrotnej kolejności.

# 2. Opis badanych algorytmów

## 2.1 Quicksort

Sortowanie szybkie stosuję metodę dziel i zwyciężaj. Jest to rekurencyjny algorytm sortowania. Polega ono na wyborze ze zbioru elementów jeden element rozdzielający, następnie dzieli się zbiór danych na dwa mniejsze zbiory, do jednego przenoszone są wszystkie elementy mniejsze niż wybrany element, a do drugiego wszystkie większe. Następnie na każdym podciągu danych używa się algorytmu sortowania szybkiego. Powtarza się to dopóki podzbiór nie będzie zawierał tylko jednego elementu.

Przeciętna złożoność obliczeniowa sortowania szybkiego jest rzędu O(n log n). W najlepszym przypadku, gdy element wybrany jest medianą wszystkich elementów, złożoność jest rzędu O(n logn). Natomiast w najgorszym przypadku, gdy zawsze będziemy wybierać element najmniejszy lub największy złożoność będzie rzędu $O(n^2)$.

## 2.2 Mergesort

Sortowanie przez scalenie stosuję metodę dziel i zwyciężaj. Jest to rekurencyjny algorytm sortowania. Wyróżnia się 3 podstawowe kroki: podzielenie danych na dwie części, następnie zastosowanie algorytmu sortowania przez scalanie na każdej z tych dwóch części, i powtarzanie tych kroków aż do momentu uzyskania tablic jednoelementowych, następnie należy łączyć części tworząc posortowane tablice.

Złożoność obliczeniowa sortowania przez scalanie jest zawsze rzędu O(n log n), w przypadku średnim, pesymistycznym i optymistycznym.

## 2.3 Introsort

Sortowanie introspektywne to sortowanie polegające na połączeniu kilku algorytmach sortowania, stosując wybrany algorytm w zależności od ilości sortowanych elementów. Korzysta on z algorytmów: sortowania przez wstawianie, sortowania przez kopcowanie oraz funkcji sortowania szybkiego polegającej na wyborze elementu rozdzielającego. Ważnym elementem sortowania introspektywnego jest głębokość rekurencji (zależna od ilości sortowanych elementów). Gdy zostanie przekroczona dana ilość rekurencji zostaje wykorzystywane sortowanie przez kopcowanie, a gdy nie zostanie przekroczona liczba rekurencji i wielkość podzbioru do sortowania jest wystarczająco mała używane jest sortowanie przez wstawianie.

Złożoność obliczeniowa sortowania introspektywnego jest zawsze rzędu O(n log n), w przypadku średnim, pesymistycznym i optymistycznym.

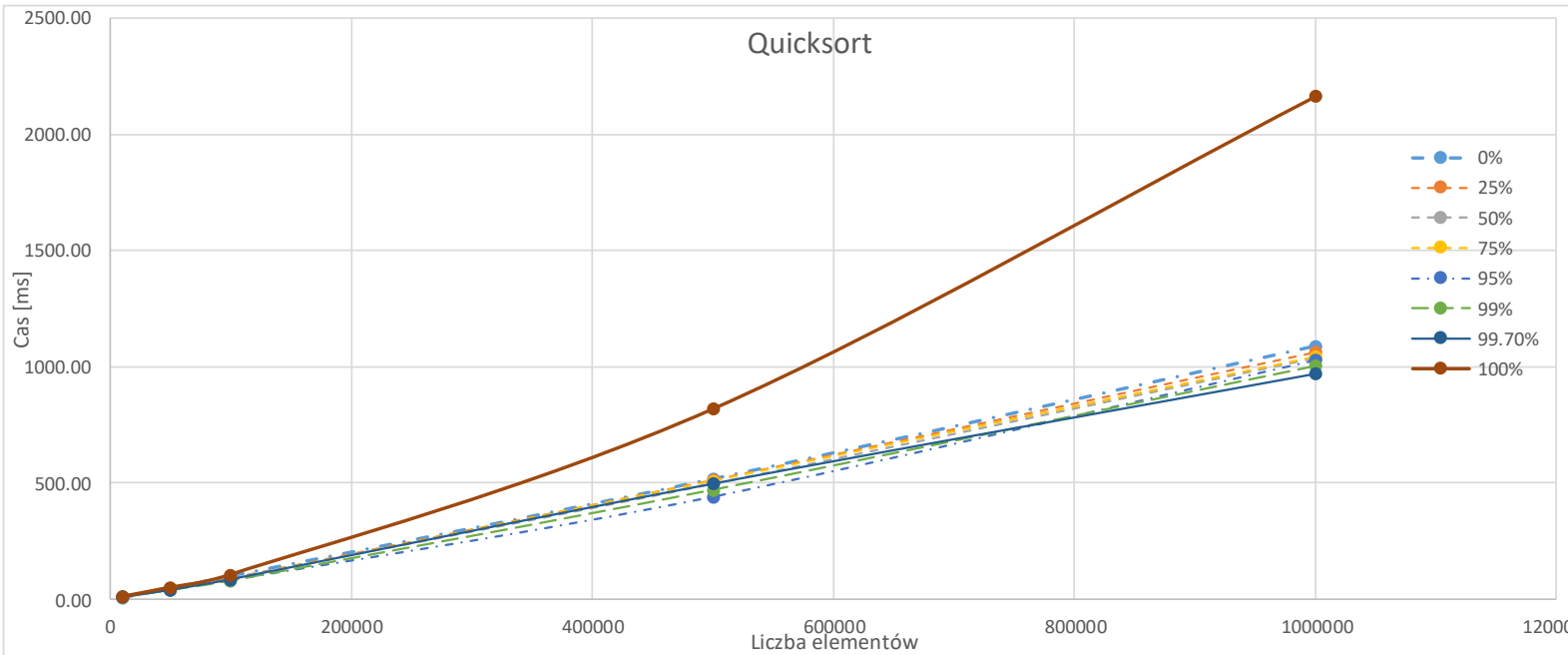
## 3. Omówienie przebiegu eksperymentów

Eksperyment polegał na liczeniu czasu sortowania każdej ze stu tablic, a następnie policzenie średniego czasu wykonania algorytmu sortowania na danym zestawie. Takie badanie czasu zostało wykonane dla każdego algorytmu sortowania, dla różnych rozmiarów zestawów danych oraz dla różnych wariantów wstępnego posortowania. Pomiar czas był wykonywany tylko dla algorytmu sortowania i nie uwzględniał innych procesów dla rzetelności wyników.
Przed sortowaniem, każdy zestaw danych był zapełniany pseudolosowymi liczbami całkowitymi, a wstępne sortowanie odbywało się za pomocą funkcji std::sort z biblioteki STL..

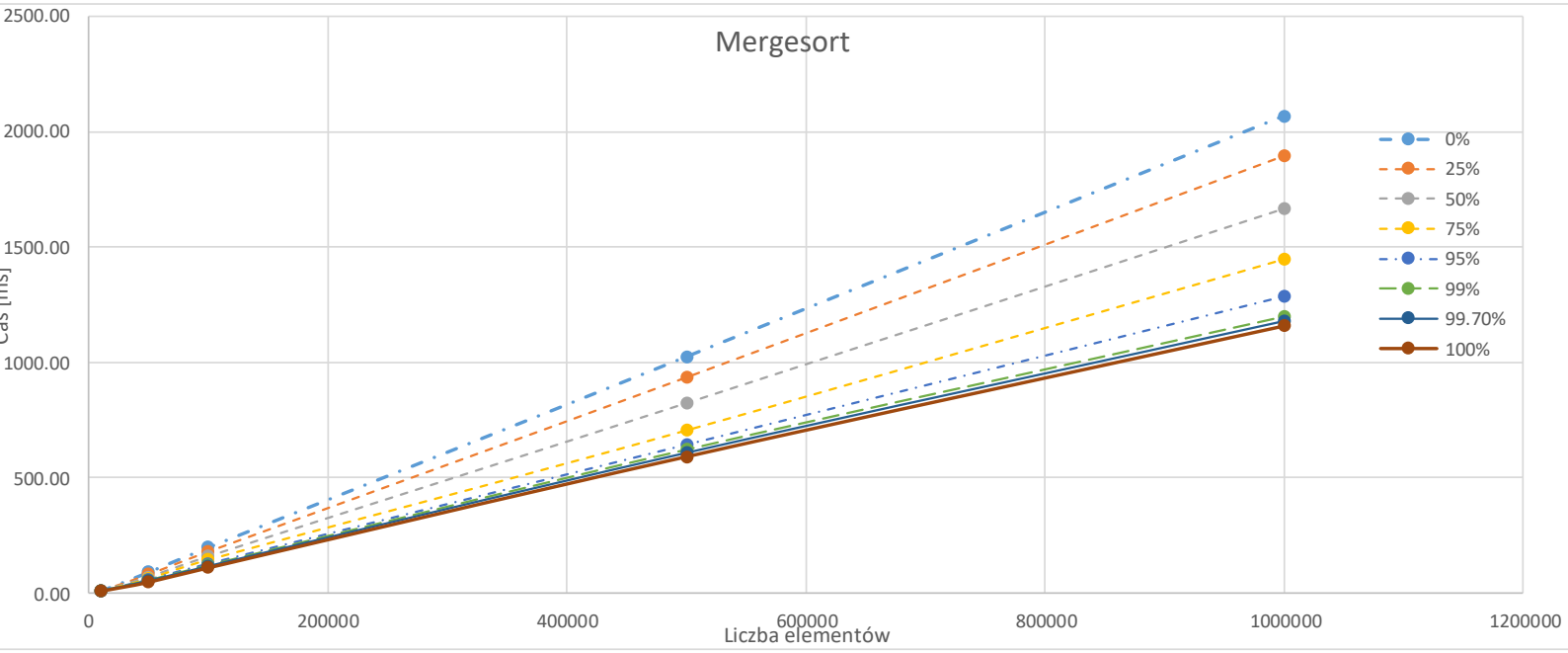
Poniżej zostały zawarte wykresy zależności średniego czasu sortowania od liczby elementów, pokazane są funkcje dla różnych wariantów wstępnego sortowania. W tabelkach zostały zawarte szczegółowe dane pokazujące średni czas sortowania w milisekundach oraz zaokrąglone do 2 miejsc po przecinku.

### 3.1 Quicksort

| Quicksort | Ilość elementów w zbiorze danych | | | | |
|---|---|---|---|---|---|
| Początkowa część zbioru już posortowana | 10000 | 50000 | 100000 | 500000 | 1000000 |
| 0% | 5.53 | 44.22 | 97.79 | 515.51 | 1087.09 |
| 25% | 7.32 | 43.57 | 88.44 | 511.44 | 1062.71 |
| 50% | 7.59 | 42.46 | 88.35 | 493.71 | 1034.78 |
| 75% | 7.40 | 40.35 | 84.02 | 507.90 | 1040.70 |
| 95% | 7.06 | 38.82 | 79.73 | 439.69 | 1028.67 |
| 99% | 7.09 | 37.63 | 78.45 | 468.38 | 1000.82 |
| 99.70% | 6.97 | 38.28 | 83.86 | 495.46 | 969.04 |
| 100% (odwrotnie) | 8.42 | 48.25 | 103.06 | 815.65 | 2161.33 |

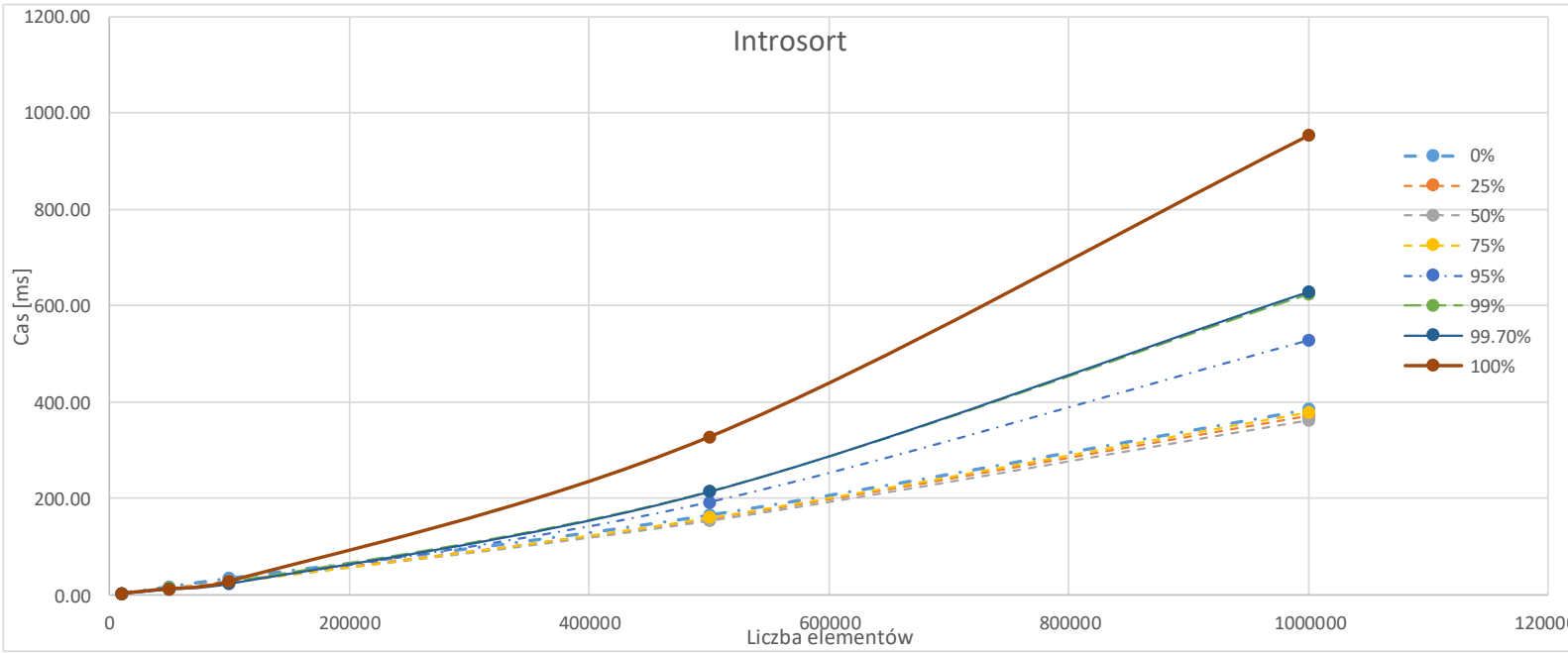


## 3.2 Mergesort




| Mergesort | Ilość elementów w zbiorze danych | | | | |
|---|---|---|---|---|---|
| Początkowa część zbioru już posortowana | 10000 | 50000 | 100000 | 500000 | 1000000 |
| 0% | 7.87 | 89.31 | 197.09 | 1023.98 | 2067.93 |
| 25% | 8.95 | 79.55 | 178.92 | 934.31 | 1894.62 |
| 50% | 8.42 | 68.45 | 158.20 | 823.31 | 1667.18 |
| 75% | 8.34 | 60.37 | 143.10 | 703.91 | 1445.84 |
| 95% | 8.23 | 56.96 | 126.32 | 643.16 | 1287.39 |
| 99% | 8.11 | 53.85 | 119.09 | 621.18 | 1198.65 |
| 99.70% | 8.30 | 51.59 | 115.67 | 607.81 | 1179.92 |
| 100% (odwrotnie) | 8.10 | 47.03 | 109.52 | 590.61 | 1156.61 |

## 3.3 Introsort




| Introsort | Ilość elementów w zbiorze danych | | | | |
|---|---|---|---|---|---|
| Początkowa część zbioru już posortowana | 10000 | 50000 | 100000 | 500000 | 1000000 |
| 0% | 2.61 | 16.25 | 33.59 | 165.16 | 385.23 |
| 25% | 2.46 | 14.40 | 28.37 | 158.01 | 371.29 |
| 50% | 2.20 | 13.26 | 26.82 | 153.96 | 363.15 |
| 75% | 2.09 | 12.30 | 25.31 | 159.83 | 379.63 |
| 95% | 2.08 | 12.55 | 25.34 | 192.36 | 528.03 |
| 99% | 1.88 | 12.89 | 25.62 | 214.22 | 624.33 |
| 99.70% | 1.89 | 11.12 | 22.68 | 213.55 | 629.01 |
| 100% (odwrotnie) | 1.91 | 11.93 | 26.82 | 326.64 | 953.56 |

## 4. Podsumowanie i wnioski

- Średnie złożoności obliczeniowe wykorzystanych algorytmów sortowania to O( n log n) i otrzymane wykresy są do tego zapisu zbliżone, aczkolwiek są podobne do funkcji liniowej, ale może to być spowodowane jedynie 5 punktami pomiarowymi lub mniejszymi tablicami niż takie, które potrzebne są do zaobserwowania funkcji O(n log n).
- Najszybszym algorytmem okazał się sortowanie introspektywne, następnie sortowanie szybkie, a najwolniejszym jest sortowanie przez scalanie.
- Poprzez wprowadzenie losowego wyboru elementu rozdzielającego spowodowaliśmy większą różnicę pomiędzy sortowaniem szybkim a przez scalanie, w jednym wypadku jest on o wiele wolniejszy (w przypadku odwrotnie posortowanej tablicy) możliwe, że losowo wybierany element był niekorzystny obliczeniowo w wielu testach spośród 100 i stąd taki wynik bardziej przybliżony do złożoności $O(n^2)$.
- Sortowanie introspektywne jest najszybsze, ponieważ wykorzystuje kilka algorytmów sortowania przez co efektywniej sortuje.

## 5. Literatura


https://www.bigocheatsheet.com/
https://en.wikipedia.org/wiki/Quicksort
https://pl.wikipedia.org/wiki/Sortowanie_szybkie
https://www.techiedelight.com/boost-quicksort-performance/
https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
https://en.wikipedia.org/wiki/Merge_sort
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://en.wikipedia.org/wiki/Introsort
https://aquarchitect.github.io/swift-algorithm-club/Introsort/
https://en.wikipedia.org/wiki/Insertion_sort