Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Sprawozdanie z projektu 1

Temat: algorytmy sortowania – przez scalanie, quicksort, introspektywne

3.04.2020

Bartosz Wójcik

249010

Kod załączony do sprawozdania: https://github.com/bartech99/PAMSI1

Prowadzący: mgr inż. Marta Emirsajłow

Grupa: piątek 7:30-9:00

1. Wprowadzenie

Założeniem zadania jest poznanie różnych algorytmów sortowania, ich właściwości oraz ich porównanie na podstawie własnego programu. Należało stworzyć po sto tablic spełniających kryteria ze względu na ilość elementów i stopień posortowania wstępnego tych tablic, każdą posortować trzema wymienionymi algorytmami oraz porównań średnie czasy sortowania jednej tablicy. Program zwraca w konsoli listę przedstawiającą średni czas posortowania jednej tablicy dla każdej kategorii na podstawie wspomnianych kryteriów, po przeprowadzeniu wszystkich prób i obliczeniu średniej. Zaimplementowane algorytmy są o złożoności $O(nlogn)$, z wyjątkiem najgorszego przypadku quicksort, który wynosi: $O(n^2)$.

2. Opis algorytmów

2.1. Quicksort (sortowanie szybkie)

Algorytm polega na stworzeniu osi podziału tablicy oraz rozdzieleniu jej na dwie partycje. W tym celu w programie zaimplementowano dwa wskaźniki – jeden na pierwszy element tablicy, drugi na ostatni oraz oś podziału tablicy przypadającą na środkowy element tablicy. Dzięki zastosowaniu rekurencji, elementy o wartości mniejszej niż wartość osi, są przenoszone na pierwszą partycję, natomiast większej – na drugą. Wyjście z tej pętli następuje w momencie „spotkania się" tych dwóch wskaźników w komórce tablicy o tym samym indeksie.

Algorytm ten jest jedyny w niniejszym zestawieniu, którego złożoność zależy od wartości elementów w przekazywanej do posortowania tablicy. W większości przypadków algorytm można uznać za wydajny, gdyż ma on złożoność $O(nlogn)$. Natomiast, jeśli wartość elementu, na który wskazuje oś podziału tablicy, ma wartość równą lub zbliżoną do wartości maksymalnej lub minimalnej spośród znajdujących się w tablicy, jego złożoność rośnie do $O(n^2)$.

2.2. Introsort (sortowanie introspektywne)

Jest to algorytm hybrydowy, co oznacza, że łączy on rozwiązania zastosowane w innych znanych algorytmach. Łączy on działanie algorytmu heapsort (sortowania przez kopcowanie), quicksort (sortowania szybkiego) oraz insertionsort (sortowania przez wstawianie). Na początku procesu, tablica jest sortowana z wykorzystaniem algorytmu szybkiego sortowania. W dalszym jego ciągu, program wywołuje sortowanie przez kopcowanie. W przypadku, gdy pozostała ilość elementów w podtablicy wyniesie mniej niż ustalony próg, są one sortowane ostatecznie za pomocą algorytmu sortowania przez wstawianie.

Dzięki takiemu połączeniu algorytmów, introsort niweluje najgorszy przypadek sortowania szybkiego i jego średnia złożoność obliczeniowa wynosi $O(nlogn)$.

2.3. Mergesort (sortowanie przez scalanie)

Algorytm ten również się opera na partycjonowaniu tablicy z liczbami do posortowania. Z wykorzystaniem rekurencji, algorytm ten dzieli tablice do momentu uzyskania samych jednoelementowych tablic. Następnie, tak podzielone elementy są scalane i jednocześnie sortowane, do utworzenia tablicy zawierającej wszystkie liczby pierwotnie przekazane do funkcji.

Jego złożoność także wynosi $O(nlogn)$.

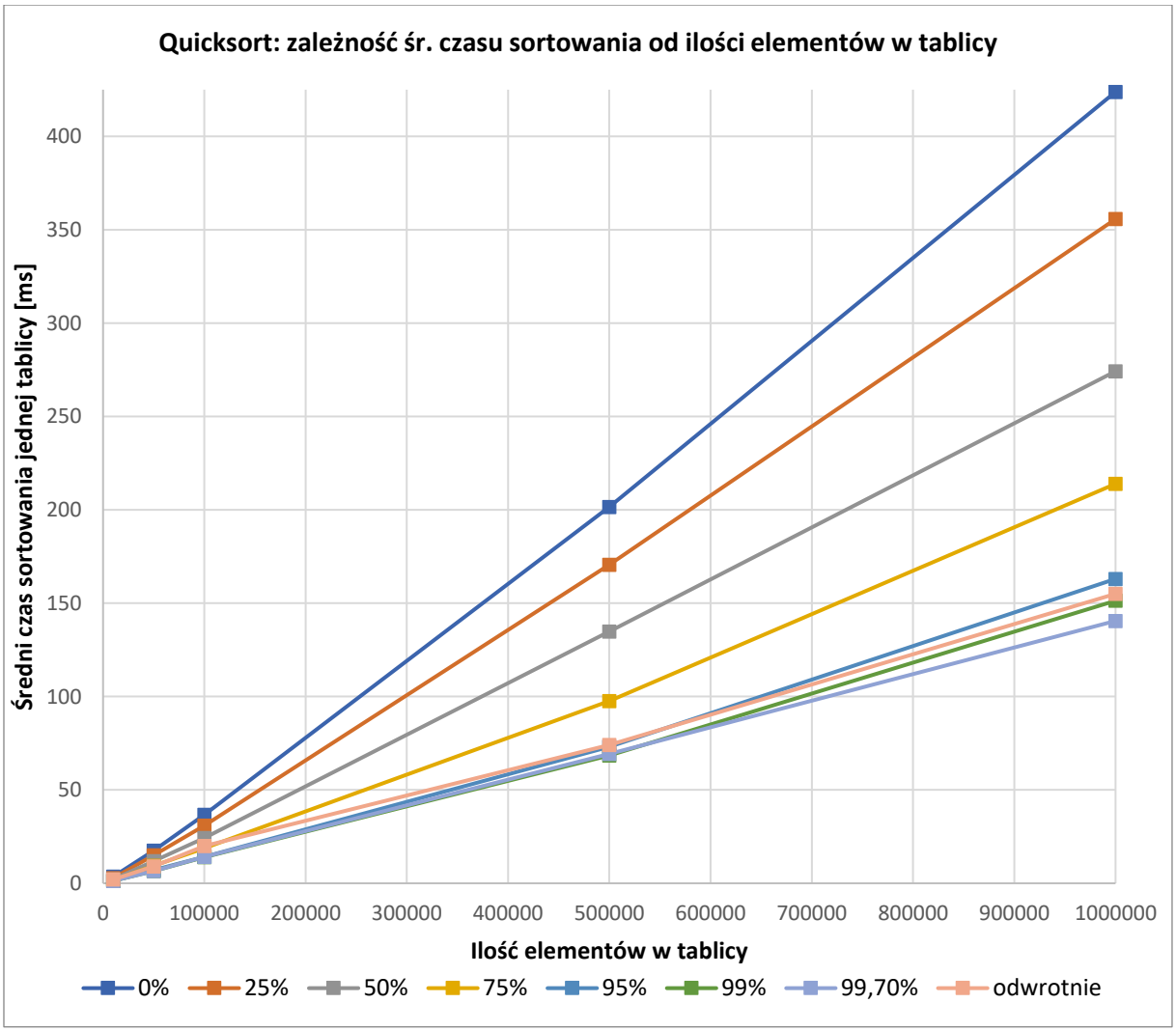
3. Opracowanie wyników

3.1. Quicksort

Wstępnie posortowane:	Ilość elementów w tablicy:				
	10000	50000	100000	500000	1000000
0%	3,5	17,4	36,6	201,5	423,8
25%	2,7	14,9	30,9	170,6	355,7
50%	2,1	11,8	24,3	134,8	274,2
75%	1,7	9,3	18,8	97,6	213,9
95%	1,3	7,1	14,1	73,0	163,0
99%	1,3	6,6	13,9	68,5	151,3
99,7%	1,3	6,5	14,2	69,3	140,5
odwrotnie	2,0	9,0	20,0	74,0	155,0

Tabela 1: Średni czas sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

* Wszystkie czasy są podane w milisekundach [ms].



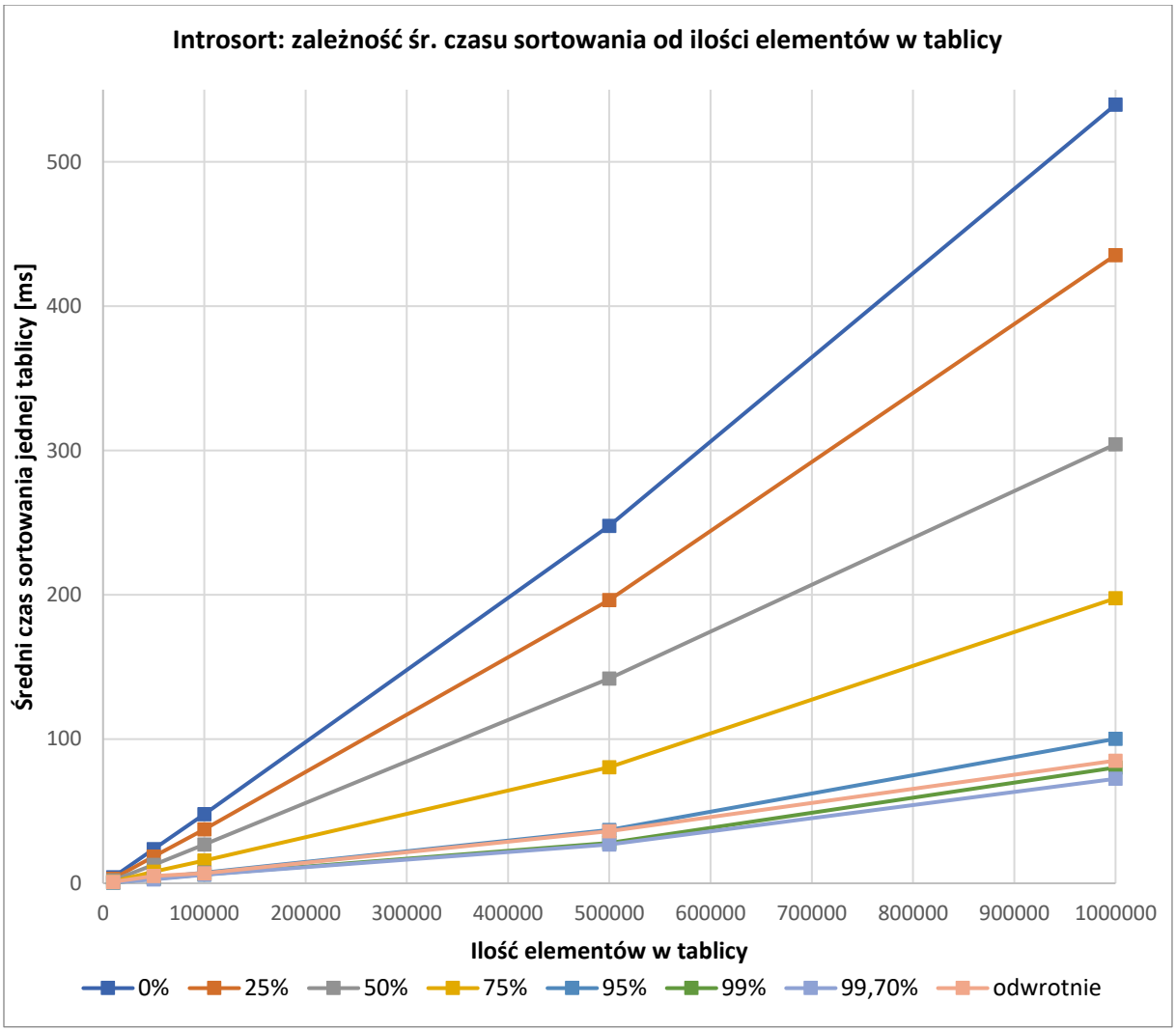
Wykres 1: Zależność średniego czasu sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

3.2. Introsort

Wstępnie posortowane:	Ilość elementów w tablicy:				
	10000	50000	100000	500000	1000000
0%	4,2	23,7	47,8	247,6	539,6
25%	3,1	18,5	37,4	196,3	435,4
50%	2,1	12,7	26,9	141,8	304,3
75%	1,2	8,1	15,8	80,4	197,5
95%	0,6	3,7	7,3	36,9	100,1
99%	0,5	3,0	6,0	27,9	80,2
99,7%	0,5	2,7	5,9	26,9	72,4
odwrotnie	1,0	5,0	7,0	36,0	85,0

Tabela 2: Średni czas sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

* Wszystkie czasy są podane w milisekundach [ms].



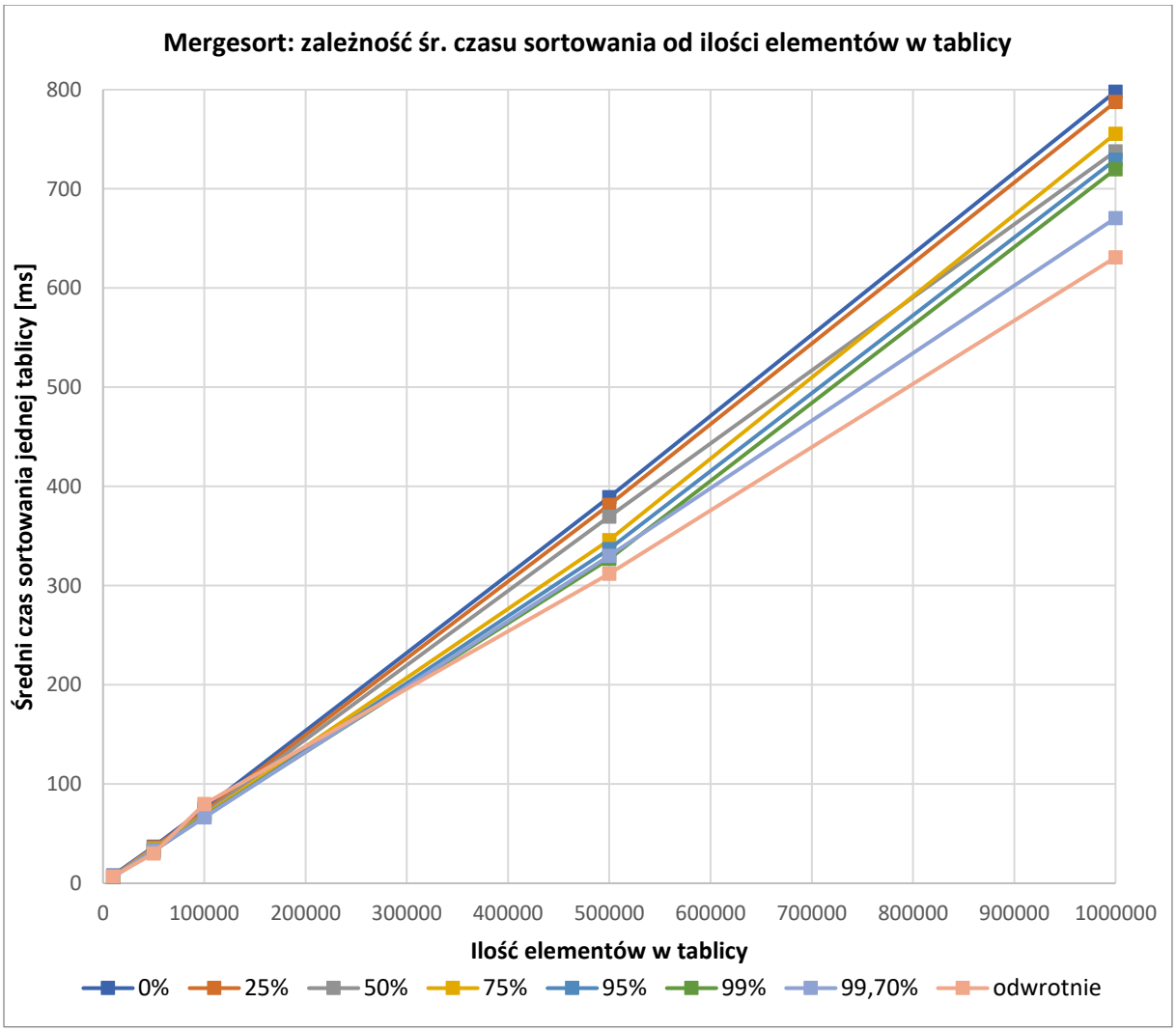
Wykres 2: Zależność średniego czasu sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

3.3. Mergesort

Wstępnie posortowane:	Ilość elementów w tablicy:				
	10000	50000	100000	500000	1000000
0%	8,0	36,8	75,1	389,2	798,1
25%	7,1	36,2	72,1	381,5	787,7
50%	6,5	35,2	69,9	369,7	737,9
75%	6,5	34,4	68,5	345,9	755,7
95%	6,4	33,3	66,6	336,9	729,5
99%	6,5	32,5	67,3	326,8	719,9
99,7%	6,8	32,9	66,4	330,1	670,6
odwrotnie	7,0	30,0	80,0	312,0	631,0

Tabela 3: Średni czas sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

* Wszystkie czasy są podane w milisekundach [ms].



Wykres 3: Zależność średniego czasu sortowania jednej tablicy przy próbie stu tablic w zależności od stopnia wstępnego posortowania oraz ilości elementów w tablicy.

4. Wnioski

Po przeanalizowaniu sporządzonych wykresów oraz danych wynika, że algorytm sortowania przez scalanie trwa względnie najdłużej.

Szybkie sortowanie zajmuje porównywalnie dużo czasu, co sortowanie introspektywne. Ten pierwszy zyskuje przewagę dla tablic posortowanych wstępnie w większym stopniu. Fakt ten wynika z pewnego stopnia losowości wartości, na którą wskazuje oś podziału tablicy w obu tych algorytmach – znajduje się ona zawsze po środku tablicy. W zależności od stopnia posortowania tablicy, posortowane wcześniej elementy znajdują się albo w jednej podtablicy w całości (jeśli stopień posortowania wstępnego jest mniejszy lub równy 50%), albo w obu (jeśli stopień ten jest większy niż 50%). Właśnie z tego powodu linie na wykresach dla quicksort i Introsort coraz szybciej „odbiegają od siebie" i wyniki mogą być niemiarodajne. Natomiast wartości elementów w tablicach nie mają zbytniego wpływu jedynie na sortowanie przez scalanie.

W programie stworzono dwie klasy – Array odpowiedzialną za przechowywanie i działania na tablicach oraz Test – zajmuje się mierzeniem czasu. Każdy z trzech algorytmów i funkcje pomocnicze znajduje się w osobnym pliku nagłówkowym.

5. Bibliografia

- https://en.wikipedia.org/wiki/Quicksort
- https://en.wikipedia.org/wiki/Introsort
- https://www.geeksforgeeks.org/introsort-or-introspective-sort/
- https://www.geeksforgeeks.org/merge-sort/
- https://www.geeksforgeeks.org/heap-sort/
- https://www.geeksforgeeks.org/quick-sort