Projektowanie algorytmów i metody sztucznej inteligencji
Projekt 2 - Algorytmy sortujące - sprawozdanie

Igor Zieliński 248944
Prowadzący: mgr inż. Marcin Ochma

Wtorek $15^{15} - 16^{55}$

1 Wstęp

Przedmiotem projektu jest implementacja oraz badanie złożoności obliczeniowej wybranych algorytmów sortującym. Algorytmy przetestowane w ramach projektu to:

- mergesort
- quicksort
- introsort

2 Opis algorytmów

2.1 Mergesort

Algorytm dzieli sortowaną tablice rekurencyjnie na dwie równe części, aż do uzyskania tablic jednoelementowych - posortowanych z definicji. Następnie dwie posortowane podtablice scalane są do jednej tablicy posortowanej.

złożoność obliczeniowa wg źródeł:

w przypadku średnim - $O(n \cdot log(n))$

w przypadku pesymistycznym - $O(n \cdot log(n))$

2.2 Quicksort

Algorytm wybiera element dzielący p (pivot), reorganizuje elementy tablicy tak, że elementy mniejsze, bądź równe p, znajdują się przed elementem p, natomiast elementy większe, za elementem p. Kolejnym krokiem jest wywołanie rekurencyjne algorytmu dla dwóch części tablicy: przed elementem p i po elemencie p.

złożoność obliczeniowa wg źródeł:

w przypadku średnim - $O(n \cdot log(n))$

w przypadku pesymistycznym - $O(n^2)$

2.3 Introsort

Algorytm łączy trzy algorytmy sortujące: heapsort, quicksort oraz insertion sort. Algorytm zapamiętuje głębokość rekurencji. Gdy jest ona mniejsza od określonego maksimum, wykonywany jest algorytm quicksort. Gdy zostanie przekroczona, wykonywany jest algorytm heapsort. Z kolei, gdy wielkość fragmentu tablicy, który jest w danym momencie sortowany jest mniejszy niż pewna przyjęta, niewielka wartość, wykonywany jest algorytm insertion sort.

złożoność obliczeniowa wg źródeł:

w przypadku średnim - $O(n \cdot log(n))$

w przypadku pesymistycznym - $O(n \cdot log(n))$

3 Opis przebiegu eksperymentu

1. Implementacja algorytmów.
2. Testy poprawności.
3. Wygenerowanie tablic zgodnie z instrukcja projektu.
4. Pomiar czasu działania algorytmów na tablicach testowych i uśrenienie wyników.

4 Wyniki eksperymentu

Poniżej zamieszczam tabele z uśrednionymi czasami sortowań oraz wykresy porównujące te czasy z funkcją $n \cdot log_2(n)$ przemnożoną przez stałą o wartości $a = 0,000025$.

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	5,116422	5,007096	3,579424	3,321928095
50000	22,685988	22,191179	14,42752	19,51205059
100000	50,506751	45,152461	30,095355	41,52410119
500000	263,18595	233,30744	160,25289	236,6446071
1000000	500,92047	569,2829	344,02854	498,2892142

Rysunek 1: Tablice losowe

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	4,6457463	4,8214081	3,546116	3,321928095
50000	21,766567	21,31595	13,836361	19,51205059
100000	48,21256	44,224561	28,936715	41,52410119
500000	254,27842	230,53983	158,51765	236,6446071
1000000	476,67305	554,98482	337,7179	498,2892142

Rysunek 2: Tablice posortowane w 25%

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	4,2715822	4,5713843	2,9770336	3,321928095
50000	20,790814	20,917082	13,303856	19,51205059
100000	46,869001	43,165073	27,913505	41,52410119
500000	241,90431	226,21088	151,77334	236,6446071
1000000	458,46738	540,23561	323,50763	498,2892142

Rysunek 3: Tablice posortowane w 50%

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	4,0189907	4,6302514	2,868481	3,321928095
50000	19,747689	20,402747	12,667139	19,51205059
100000	44,666463	41,565245	27,001179	41,52410119
500000	229,04382	215,97341	140,31253	236,6446071
1000000	434,06385	537,19865	319,18348	498,2892142

Rysunek 4: Tablice posortowane w 75%

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	3,87883	4,6071611	2,7452546	3,321928095
50000	19,047673	20,201428	12,097243	19,51205059
100000	44,034819	40,175056	25,646357	41,52410119
500000	221,15672	213,68989	135,04653	236,6446071
1000000	414,68997	516,10226	296,39269	498,2892142

Rysunek 5: Tablice posortowane w 95%

rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	3,9031279	4,5242138	2,6392882	3,321928095
50000	18,903023	20,561806	12,180118	19,51205059
100000	43,98518	39,954557	25,172158	41,52410119
500000	217,72027	214,47251	135,4804	236,6446071
1000000	404,50881	515,35141	294,60296	498,2892142

Rysunek 6: Tablice posortowane w 99%

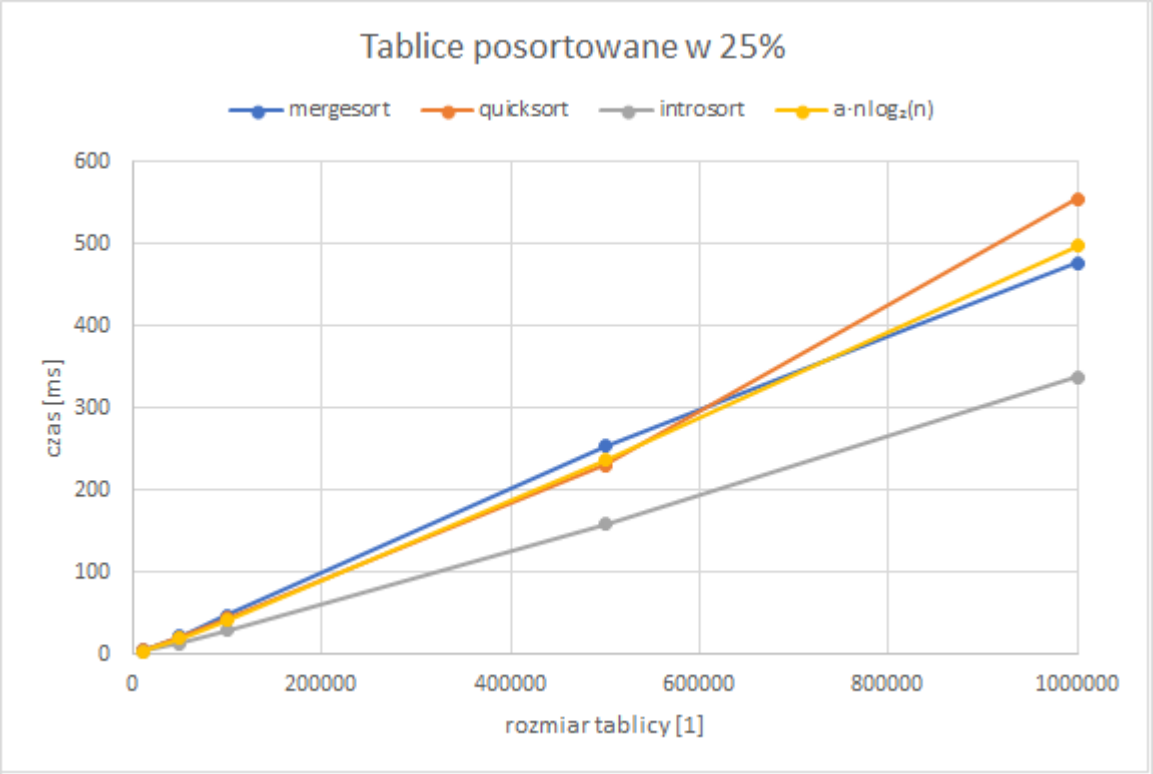
rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	3,865542	4,5844483	2,6562062	3,321928095
50000	18,811514	20,123639	11,990136	19,51205059
100000	43,384266	40,695358	25,941658	41,52410119
500000	211,84521	211,59336	135,73963	236,6446071
1000000	405,38561	522,09673	294,31911	498,2892142

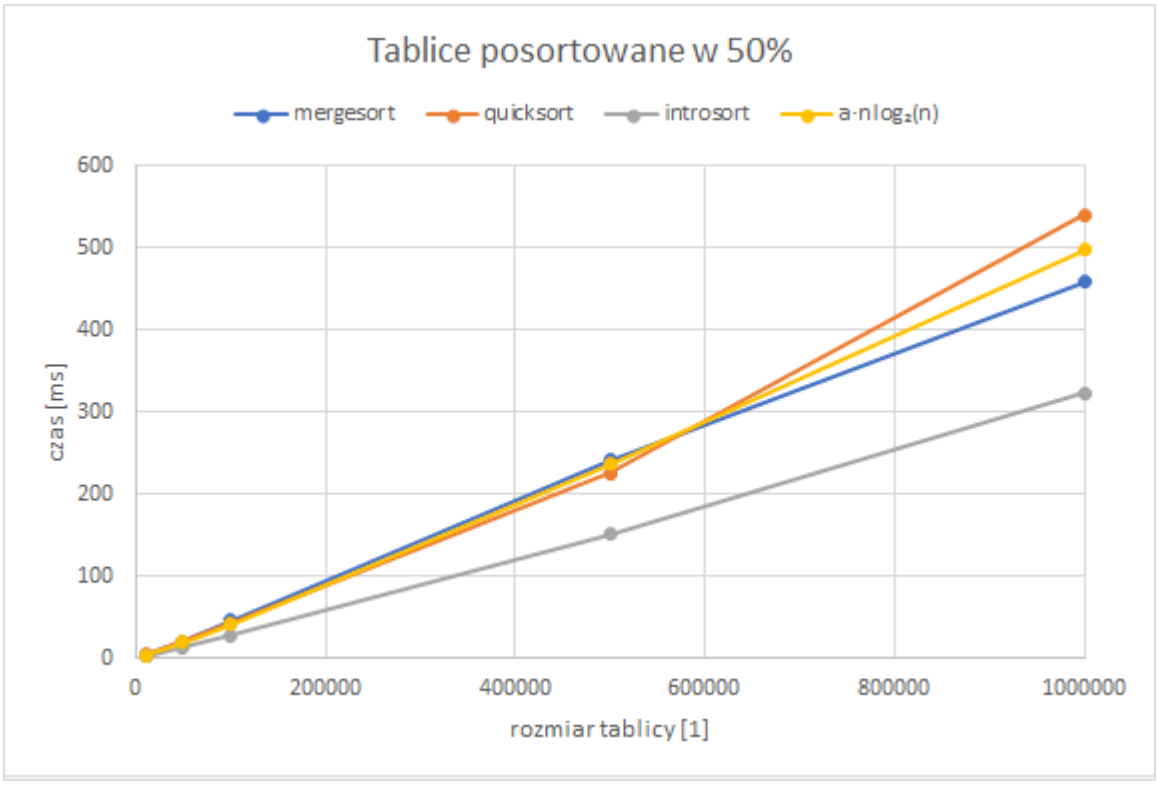
Rysunek 7: Tablice posortowane w 99,7%

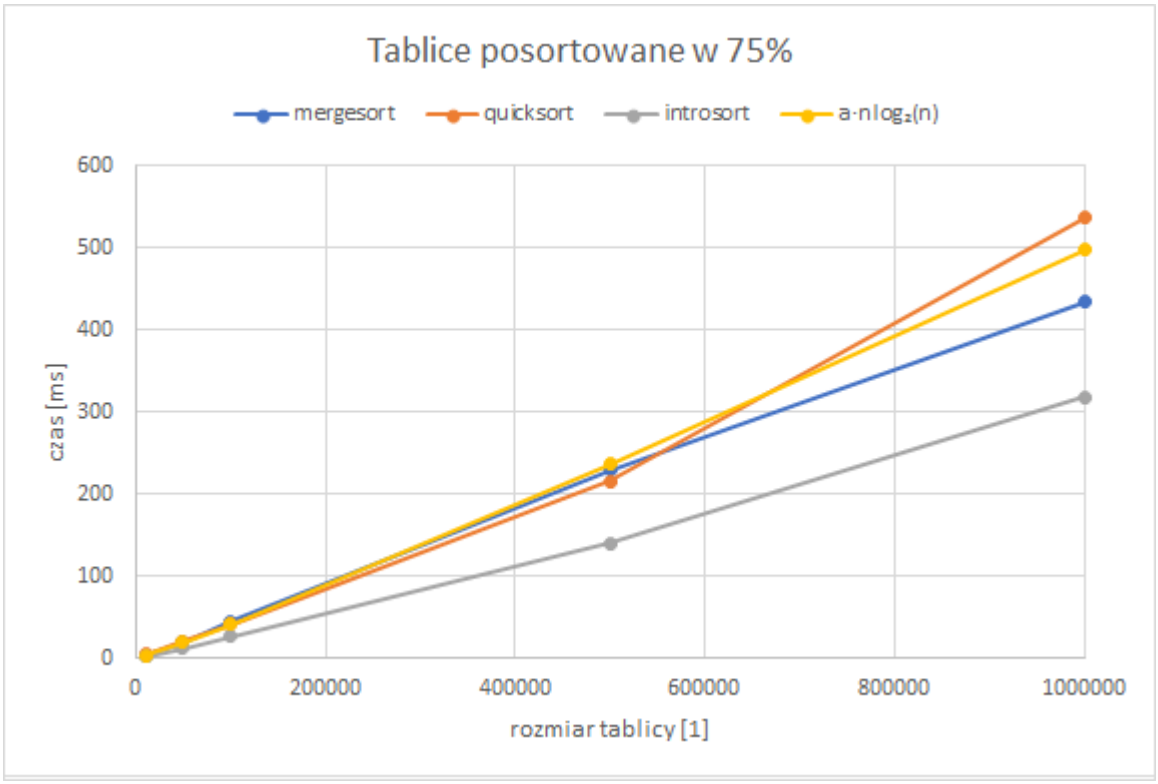
rozmiar tablicy	mergesort[ms]	quicksort[ms]	introsort[ms]	a·nlog₂(n)[ms]
10000	3,9981033	4,48107	2,6126505	3,321928095
50000	19,019279	20,790013	12,016239	19,51205059
100000	43,359287	40,835155	25,833391	41,52410119
500000	218,40072	209,94968	137,057	236,6446071
1000000	414,6626	519,07385	289,34908	498,2892142

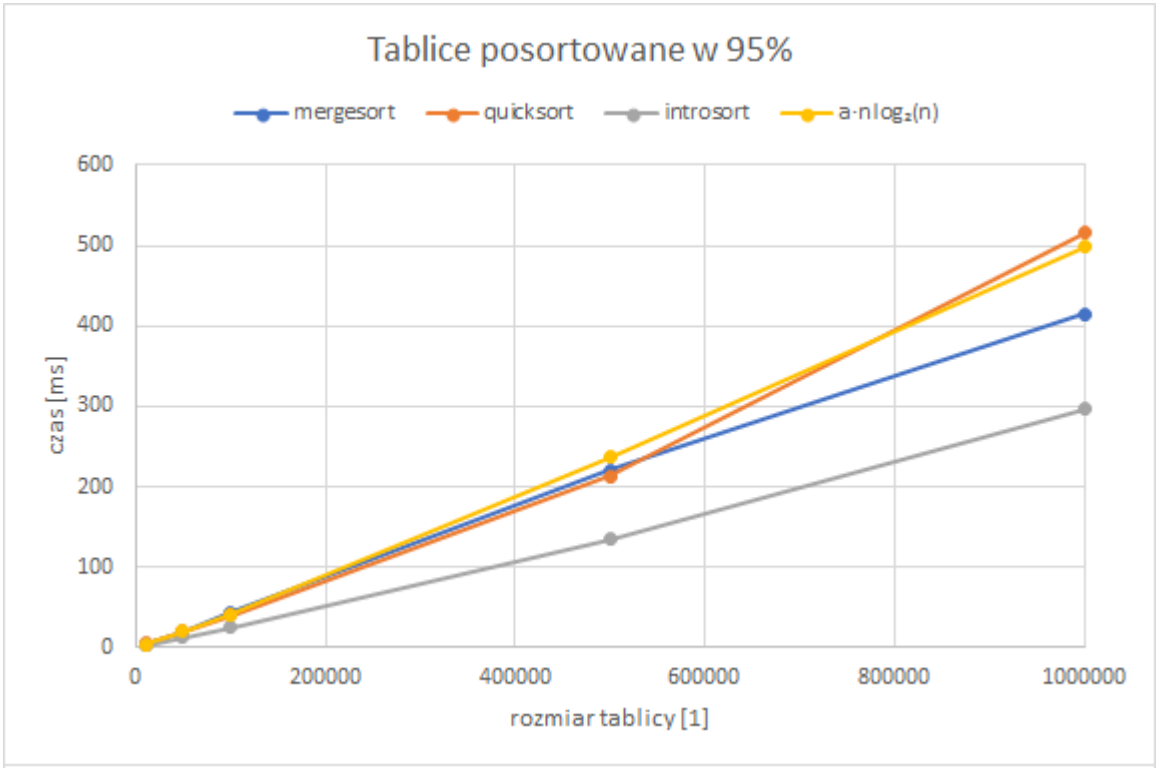
Rysunek 8: Tablice posortowane malejąco

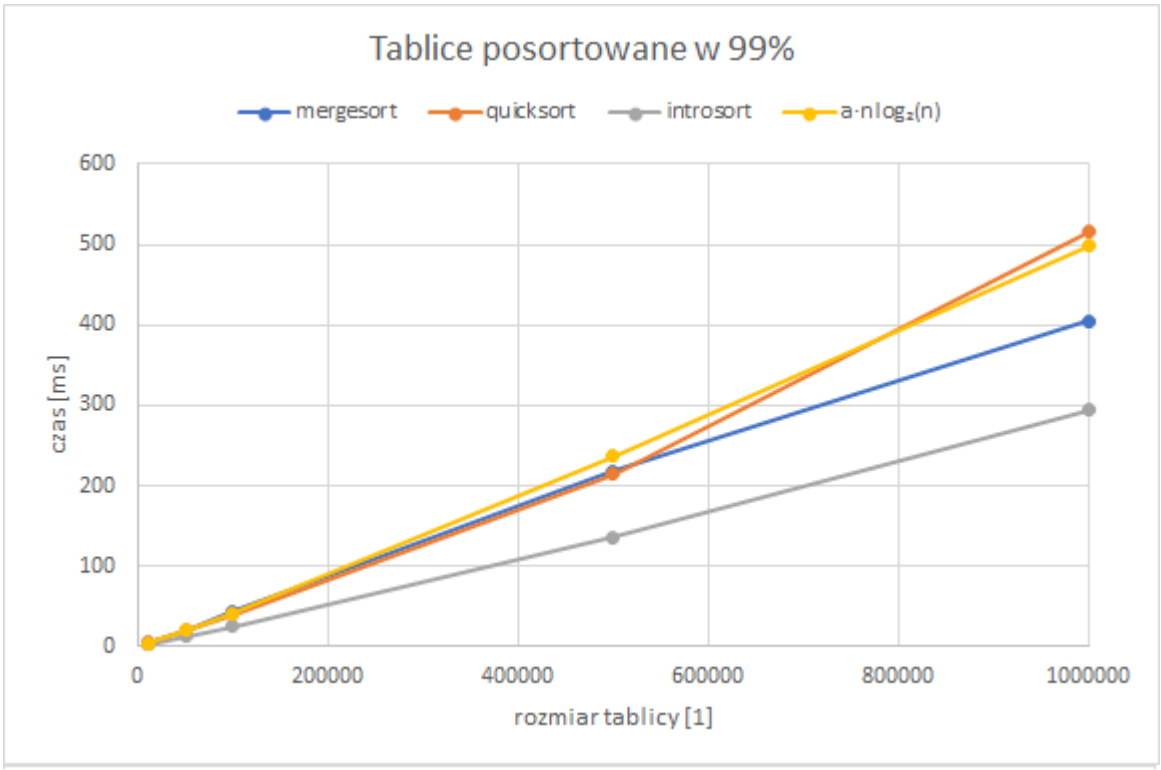

Tablice losowe
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas[ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]

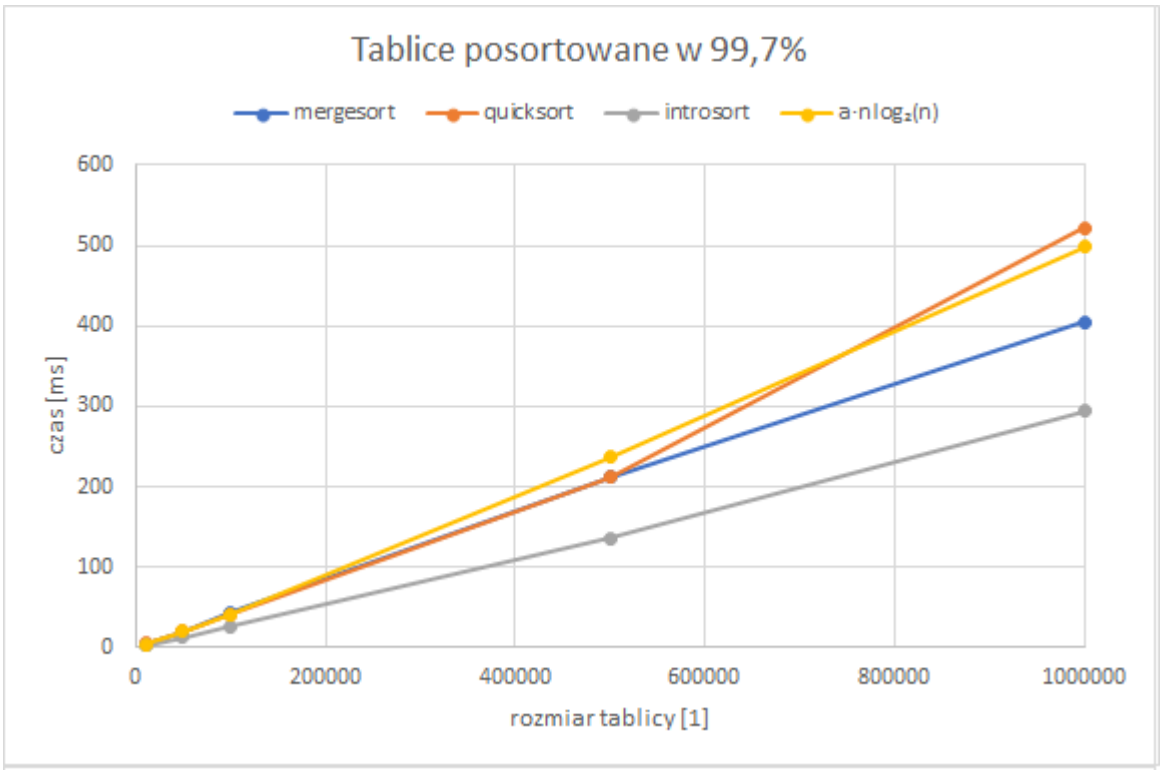

Tablice posortowane w 25%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]

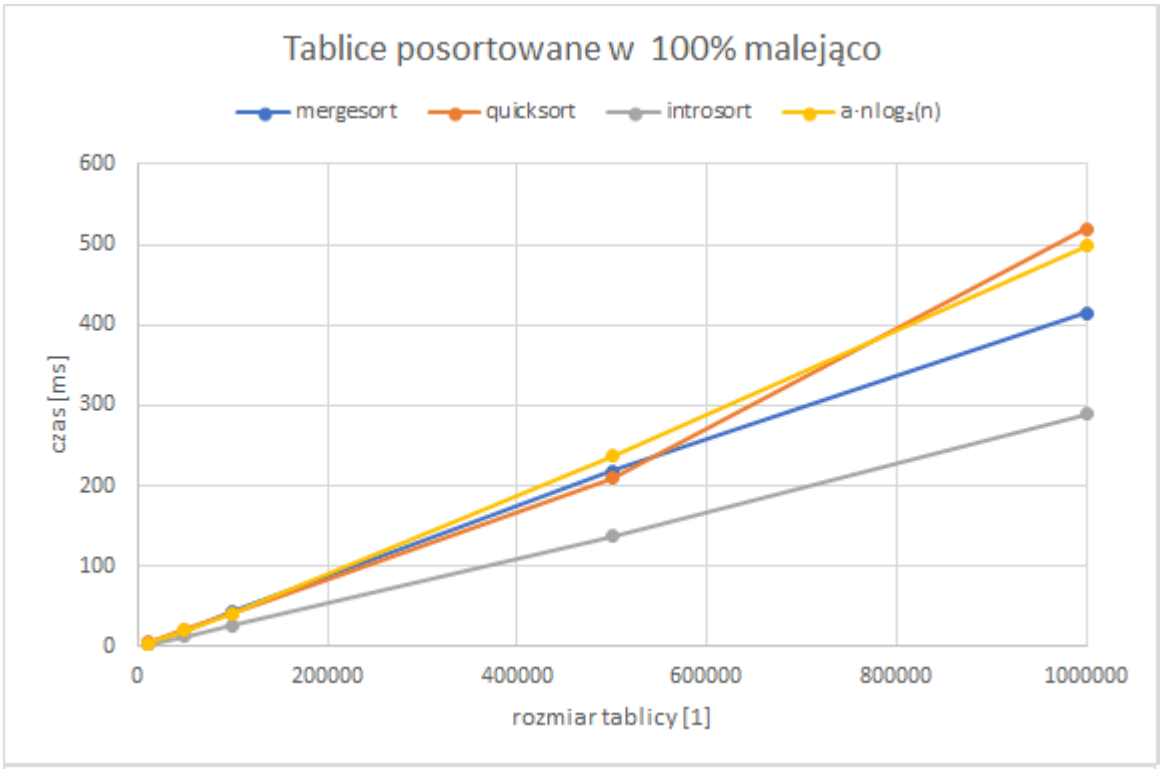

Tablice posortowane w 50%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]


Tablice posortowane w 75%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]


Tablice posortowane w 95%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]


Tablice posortowane w 99%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]


Tablice posortowane w 99,7%
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]


Tablice posortowane w 100% malejąco
mergesort
quicksort
introsort
a·nlog₂(n)
600
500
400
300
200
100
0
czas [ms]
0
200000
400000
600000
800000
1000000
rozmiar tablicy [1]

5 Wnioski

1. Jak widać na załączonych wykresach, dla średniego przypadku wszystkie z badanych algorytmów znajdują się w klasie złożoności obliczeniowej $O(n \cdot log_2(n))$, więc potwierdza to informacje ze źródeł.

2. Wraz ze wzrostem uporządkowania tablic wejściowych, obserwujemy spadek średniego czasu działania algorytmów. Wyjątkiem jest sytuacja dla $99,7\%$ uporządkowania dla algorytmu quicksort. Różnica ta że może wynikać ze specyficznych rodzajów testów i zbyt małej ich ilości oraz zwyczajnie "pecha", gdyż w mojej wersji quicksort pivot wybierany jest losowo. Powodem obniżenia czasu działania w częściowo uporządkowanych zbiorach jest mniejsza ilość operacji na elementach tablic.

6 Źródła

https://pl.wikipedia.org/wiki/Sortowanie_szybkie
https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://pl.wikipedia.org/wiki/Sortowanie_przez_wstawianie