Filip Materak

249482

Piątek 13:15-15:00

E08-06j

mgr Marta Emirsajłow

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 1
Algorytmy sortujące

1 Wstęp

Tematem projektu było zaimplementowanie następujących algorytmów sortujących: przez scalanie, sortowanie szybkie oraz sortowanie introspektywne. Wszystkie w.w. algorytmy należą do grupy sortowań zaawansowanych, tzn. o logarytmicznej złożoności obliczeniowej.
Wyjątkiem jest tylko pesymistyczny przypadek quicksortu, którego złożoność obliczeniowa wynosi $O(n^2)$.
Sortowanie przez scalanie jako jedyne z tej grupy jest sortowaniem stabilnym, oznacza to, że występujące po sobie elementy o jednakowej wartości, nie zostaną zamienione ze sobą miejscami w procesie sortowania.

W celu porównania algorytmów, należało przeprowadzić pomiary czasu sortowania tablic 10.000, 50.000, 100.000, 500.000 i 1.000.000 elementowych, o różnych stopniach wstępnego posortowania tj. $0\%, 25\%, 50\%, 75\%, 95\%, 99\%, 99.7\%$ oraz tablicy posortowanej odwrotnie.

2 Sortowanie przez scalanie

Sortowanie przez scalanie jest algorytmem rekurencyjnym. Tablica w każdym kroku zostaje podzielona na dwie części, aż do powstania tablic jednoelementowych, które są już de facto posortowane. Po wystąpieniu przypadku bazowego, algorytm porównuje obie tablice, a następnie scala je, układając ich elementy w odpowiedniej kolejności.

Ponieważ merge-sort w każdym wywołaniu rekurencji dzieli tablicę na pół, to można zauważyć że maksymalna głębokość rekurencji jest równa wysokości kompletnego drzewa binarnego tj. $log_2 n$. Następnie dla każdego elementu wykonywane jest porównanie i scalenie, więc ostateczna złożoność obliczeniowa wynosi $O(n \cdot log_2 n)$. Rozpatrując przypadek najgorszy można zauważyć, że zwiększa się jedynie ilość potrzebnych porównań podczas scalania elementów, więc nie powoduje to zmiany złożoności obliczeniowej.

3 Sortowanie szybkie

Sortowanie szybkie również jest algorytmem rekurencyjnym. W każdym kroku sortowania szybkiego zostaje wybrany element służacy do podziału tablicy. Następnie algorytm porównuje wszystkie elementy tablicy z wybranym i tworzy 2 nowe tablice, jedną zawierającą elementy mniejsze, a drugą zawierającą większe. Element wybrany do podziału nie bierze dalej udziału w sortowaniu, ponieważ jest już na swojej pozycji. Kroki te są powtarzane aż do uzyskania posortowanej tablicy.

Dla każdego kroku algorytm wykonuje n porównań. Złożoność obliczeniowa zależy od wyboru elementu rozdzielającego. Średnio można powiedzieć, że głębokość rekurencji wynosi $logn$, więc całkowita złożoność obliczeniowa dla średniego przypadku wynosi $O(n \cdot logn)$. W pesymistycznym przypadku algorytm w każdym kroku może wybrać element największy lub najmniejszy w tablicy, co doprowadzi do głębokości rekurencji równej n, więc całkowita złożoność obliczeniowa będzie wynosić $O(n^2)$.

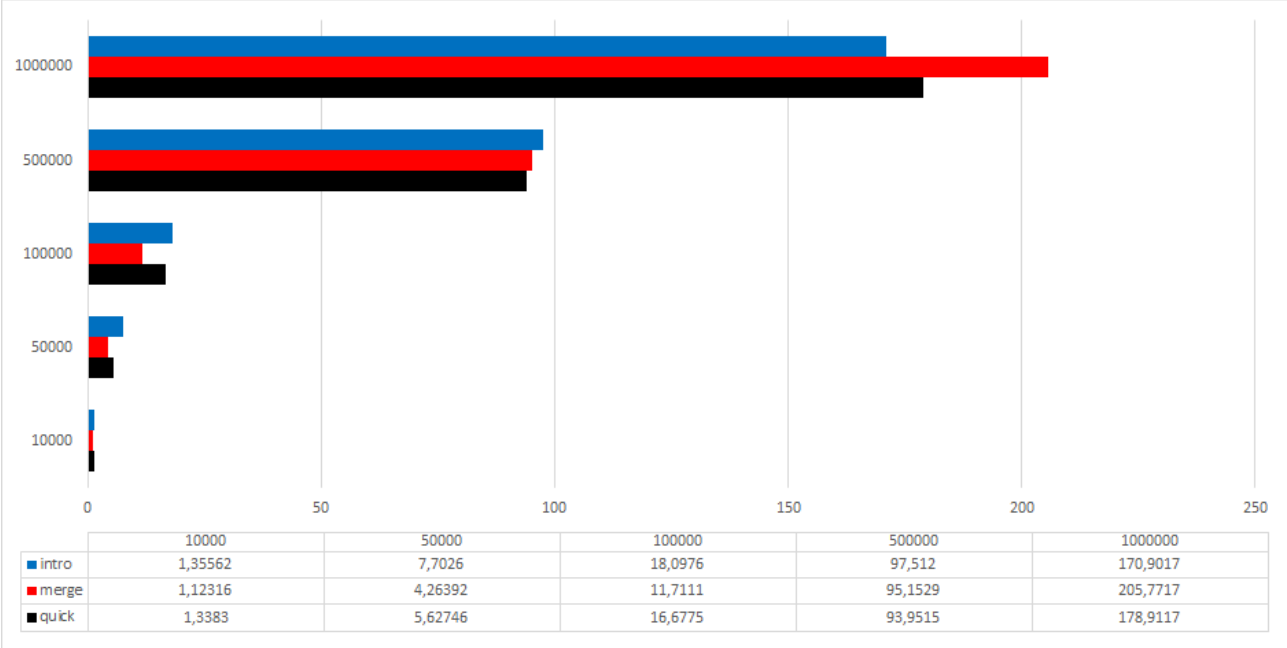
4 Sortowanie introspektywne

Sortowanie introspektywne jest sortowaniem hybrydowym, tzn. zawiera w sobie więcej niż jeden algorytm sortowania. Bazuje ono na sortowaniu szybkim i sortowaniu przez kopcowanie, i pozwala wyeliminować złożoność $O(n^2)$ dla najgorszego przypadku.

Pierwszym krokiem jest obliczenie maksymalnej głębokości rekurencji, można ją dobrać dowolnie. Następnie tablica jest dzielona jak w przypadku sortowania szybkiego, jednak przekazuje się maksymalną głębokość rekurencji jako parametr, który w każdym kroku jest dekrementowany. W momencie, kiedy wynosi on zero, algorytm uznaje otrzymaną tablicę za przygotowaną, i sortuje ją przez kopcowanie.

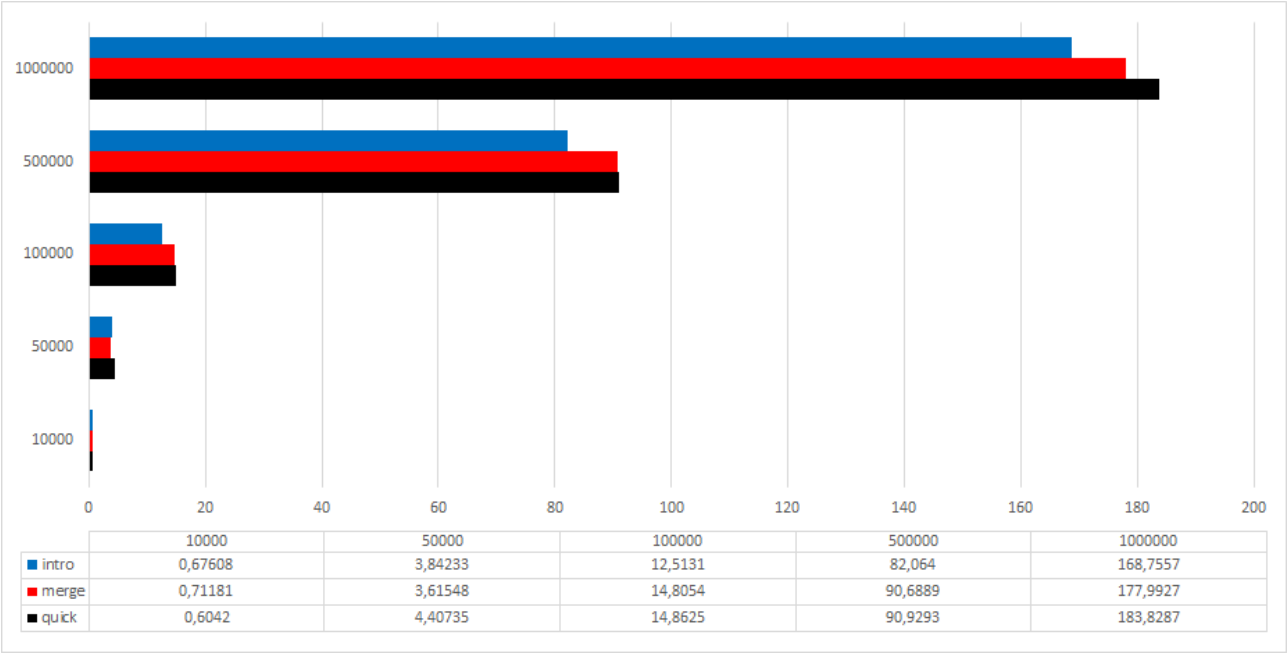
Taki algorytm pozwala na wykorzystanie najlepszych cech obu sortowań. Sortowanie szybkie dzieli tablice na małe podzbiory, które są kopcowane. Złożoność obliczeniowa kopcowania to $O(n \cdot log_2 n)$, jednak dla dużych podzbiorów jest ono kilkukrotnie wolniejsze niż sortowanie szybkie. W sortowaniu introspektywnym zostaje jednak wywoływane dla dużo mniejszych tablic, co pozwala na otrzymanie podobnych czasów sortowania, co w średnim przypadku sortowania szybkiego, z jednoczesną eliminacją najgorszego przypadku.

5 Wyniki pomiarów



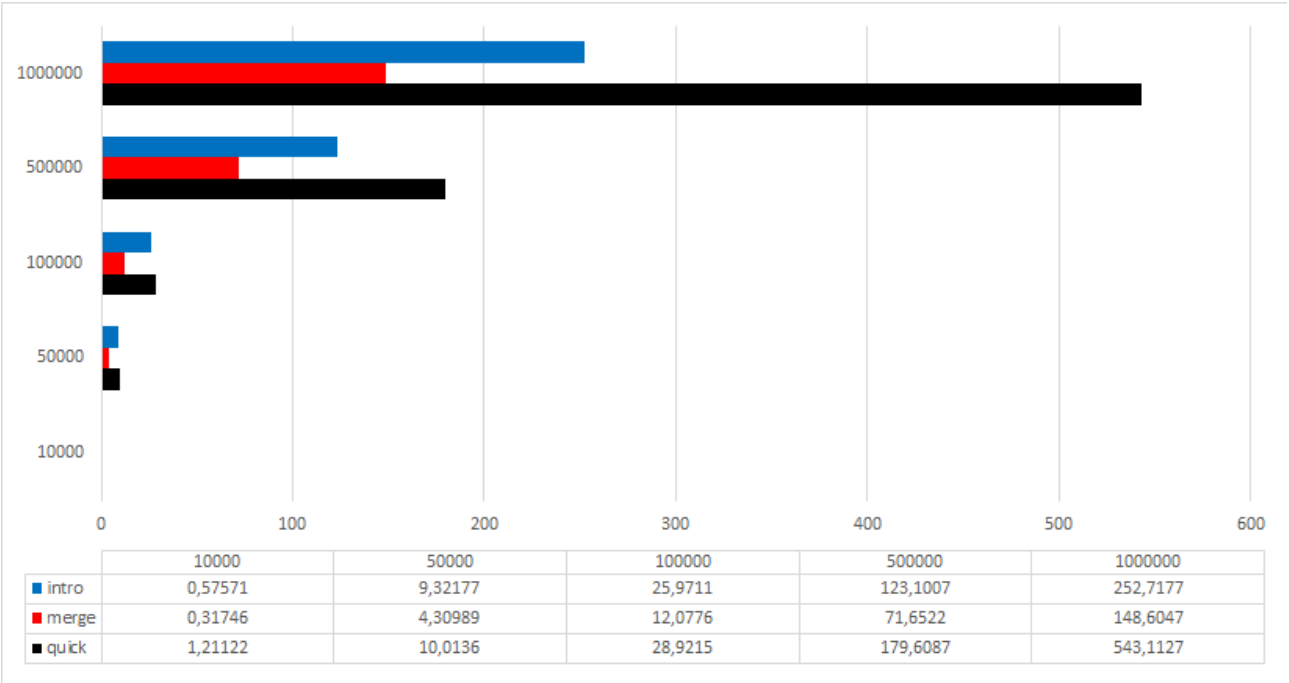
	10000	50000	100000	500000	1000000
intro	1,35562	7,7026	18,0976	97,512	170,9017
merge	1,12316	4,26392	11,7111	95,1529	205,7717
quick	1,3383	5,62746	16,6775	93,9515	178,9117

Rysunek 1: Wyniki pomiarów czasu dla tablic nieposortowanych



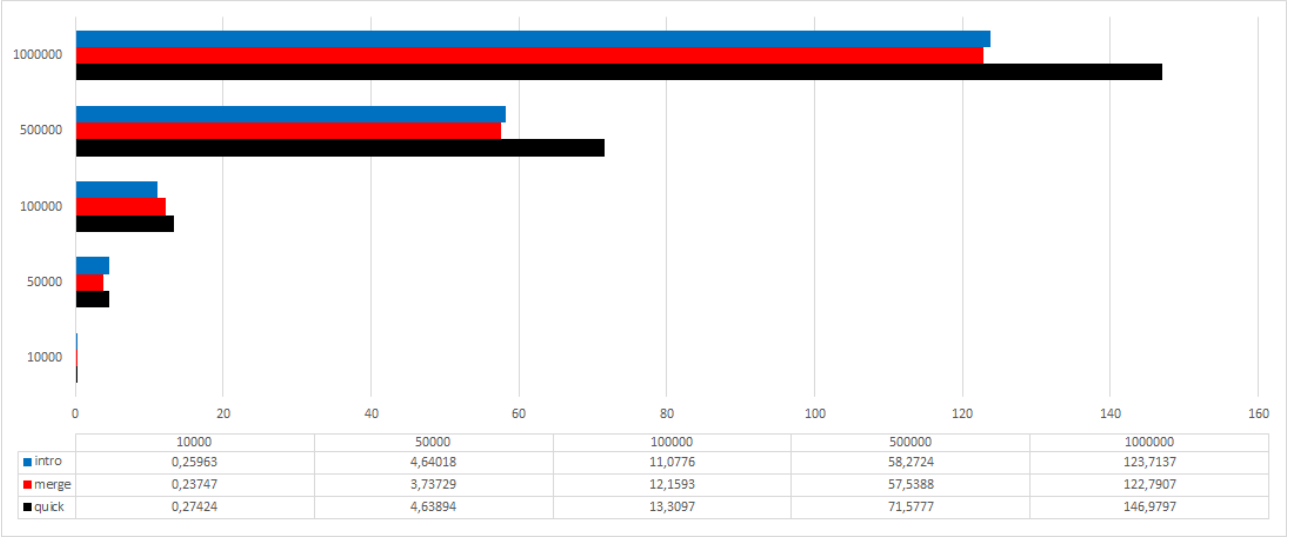
	10000	50000	100000	500000	1000000
intro	0,67608	3,84233	12,5131	82,064	168,7557
merge	0,71181	3,61548	14,8054	90,6889	177,9927
quick	0,6042	4,40735	14,8625	90,9293	183,8287

Rysunek 2: Wyniki pomiarów czasu dla tablic posortowanych w 25%



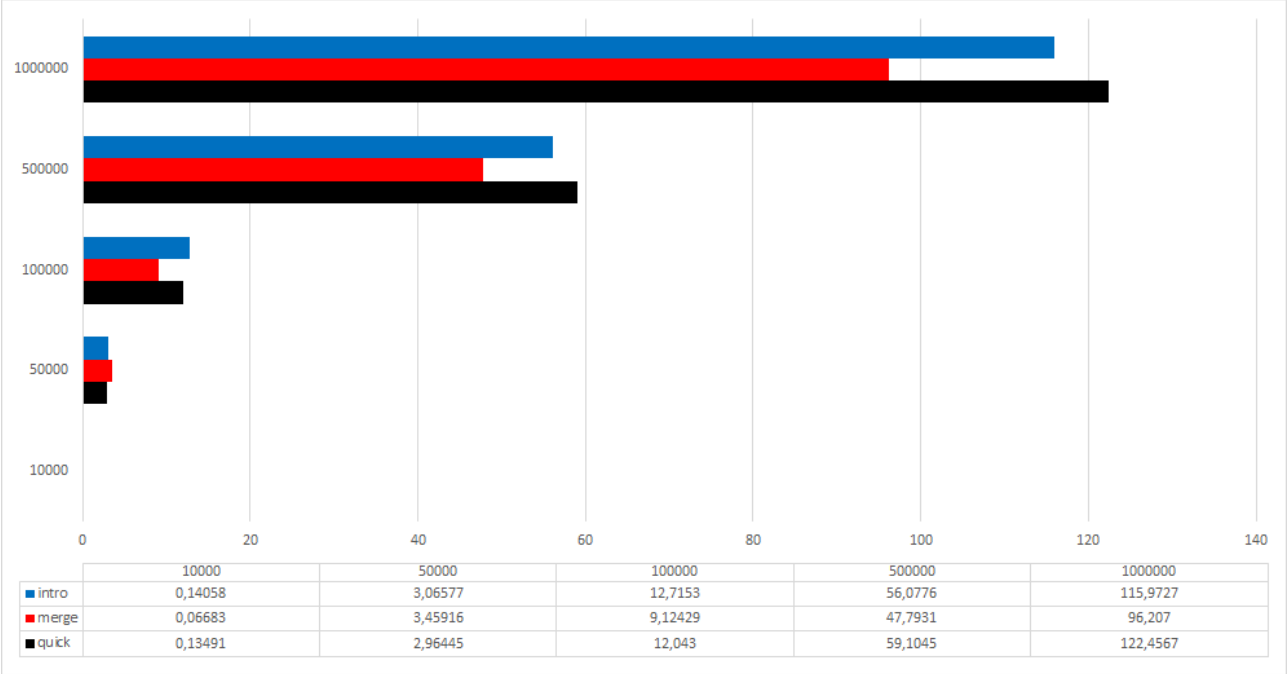
	10000	50000	100000	500000	1000000
intro	0,57571	9,32177	25,9711	123,1007	252,7177
merge	0,31746	4,30989	12,0776	71,6522	148,6047
quick	1,21122	10,0136	28,9215	179,6087	543,1127

Rysunek 3: Wyniki pomiarów czasu dla tablic posortowanych w 50%



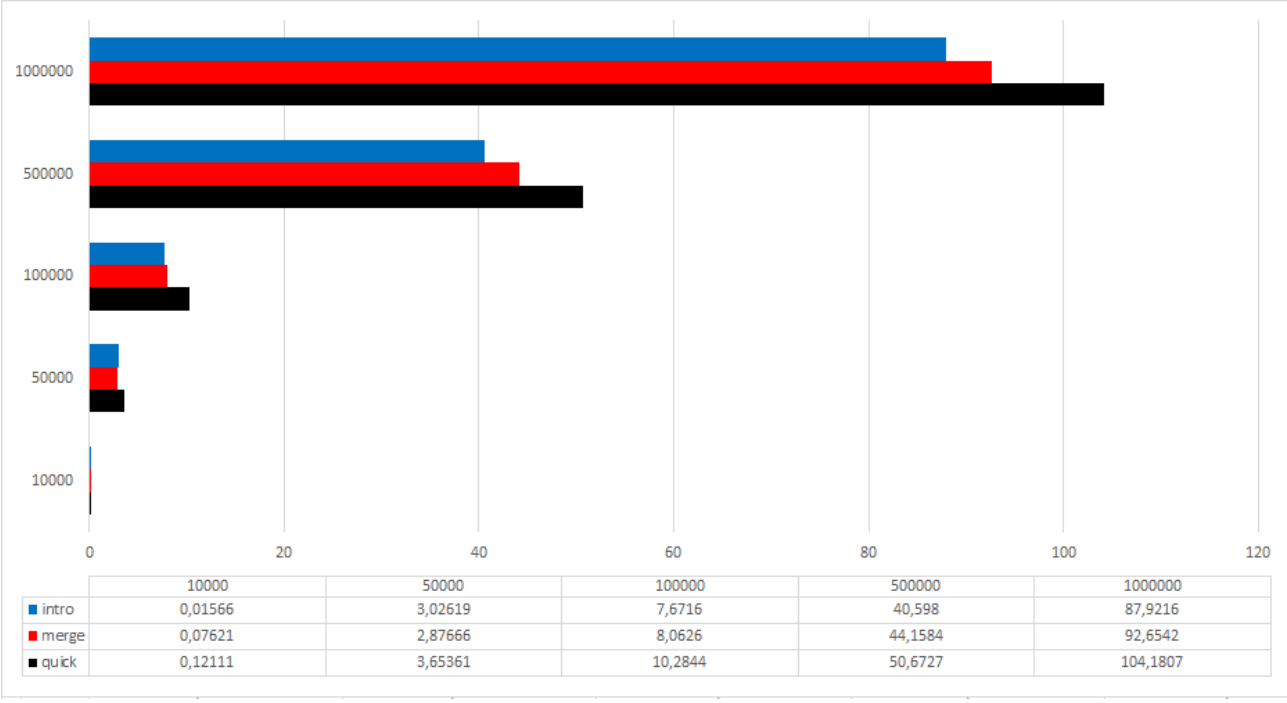
	10000	50000	100000	500000	1000000
intro	0,25963	4,64018	11,0776	58,2724	123,7137
merge	0,23747	3,73729	12,1593	57,5388	122,7907
quick	0,27424	4,63894	13,3097	71,5777	146,9797

Rysunek 4: Wyniki pomiarów czasu dla tablic posortowanych w 75%



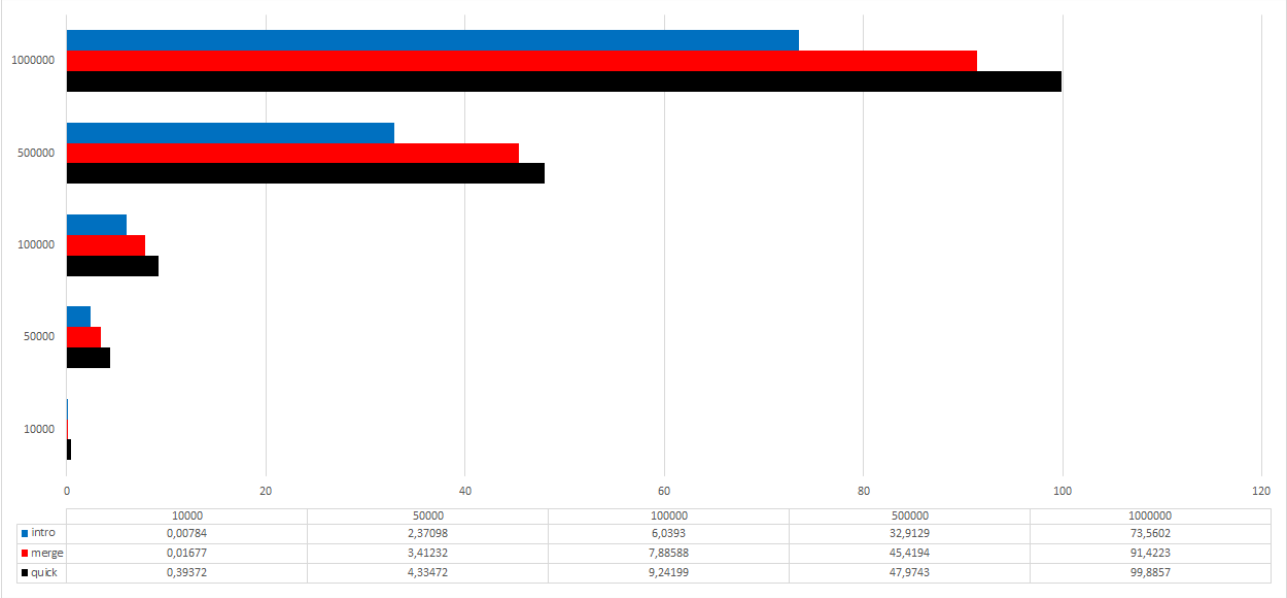
	10000	50000	100000	500000	1000000
intro	0,14058	3,06577	12,7153	56,0776	115,9727
merge	0,06683	3,45916	9,12429	47,7931	96,207
quick	0,13491	2,96445	12,043	59,1045	122,4567

Rysunek 5: Wyniki pomiarów czasu dla tablic posortowanych w 95%



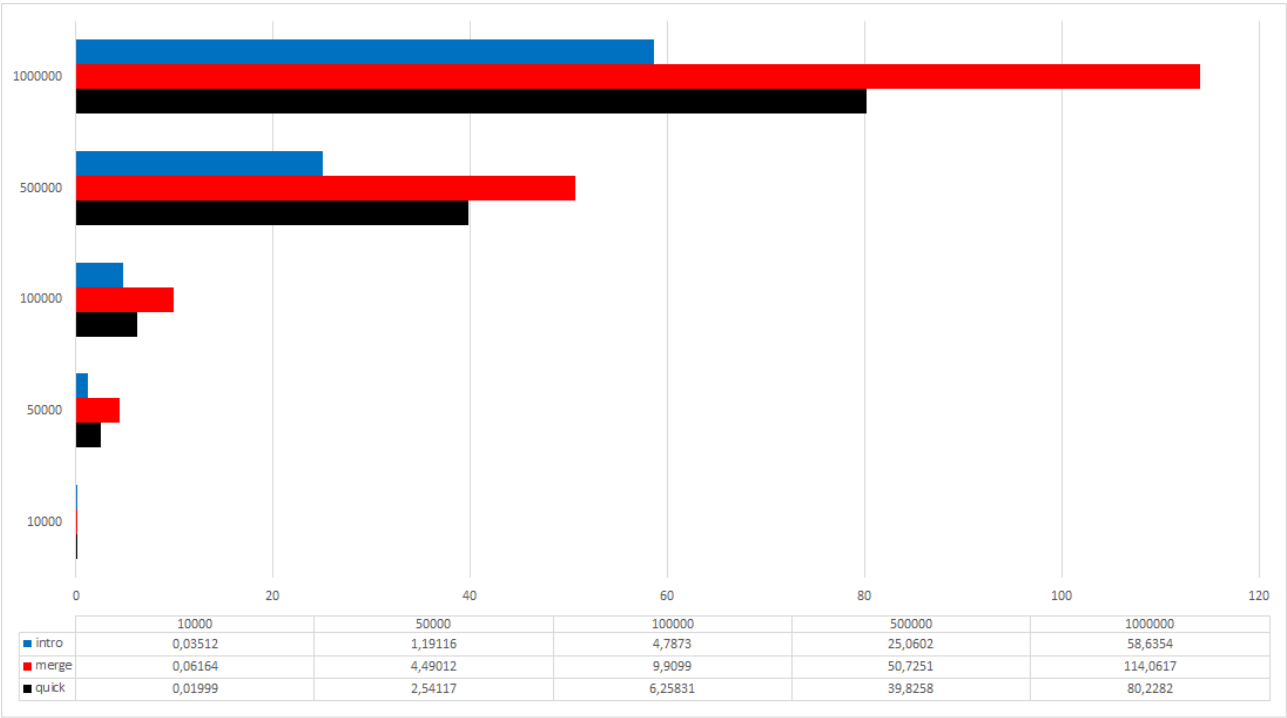
	10000	50000	100000	500000	1000000
intro	0,01566	3,02619	7,6716	40,598	87,9216
merge	0,07621	2,87666	8,0626	44,1584	92,6542
quick	0,12111	3,65361	10,2844	50,6727	104,1807

Rysunek 6: Wyniki pomiarów czasu dla tablic posortowanych w 99%



	10000	50000	100000	500000	1000000
intro	0,00784	2,37098	6,0393	32,9129	73,5602
merge	0,01677	3,41232	7,88588	45,4194	91,4223
quick	0,39372	4,33472	9,24199	47,9743	99,8857

Rysunek 7: Wyniki pomiarów czasu dla tablic posortowanych w 99,7%



	10000	50000	100000	500000	1000000
intro	0,03512	1,19116	4,7873	25,0602	58,6354
merge	0,06164	4,49012	9,9099	50,7251	114,0617
quick	0,01999	2,54117	6,25831	39,8258	80,2282

Rysunek 8: Wyniki pomiarów czasu dla tablic posortowanych odwrotnie

6 Podsumowanie

Wszystkie algorytmy są dobrze działające, jednak w wyniku testów można zauważyć specyficzne przypadki dla których algorytmy tracą na wydajności. Dla tablic posortowanych w 50% sortowanie szybkie oraz introspektywne było dużo wolniejsze niż w pozostałych przypadkach ponieważ oba algorytmy wybierają piwot w środku zbioru, podobnie sortowanie przez scalanie było wyraźnie wolniejsze dla tablicy posortowanej odwrotnie. W pozostałych przypadkach wyniki były zbliżone. Sortowanie introspektywne będące hybrydą sortowania szybkiego uniknęło ogromnego spowolnienia na tablicy posortowanej w 50% i wykazywało się stabilną szybkością we wszystkich przypadkach. Wszystkie algorytmy najlepiej radziły sobie z tablicami posortowanymi wcześniej w ponad 95%, pozwalało to skrócić czas sortowania kilkukrotnie. Przewidywane wyniki zostały w większości potwierdzone wynikami testów. Oznacza to, że algorytmy zostały zaimplementowane w poprawny sposób oraz, że testy zostały przeprowadzone w sposób prawidłowy.

7 Literatura

Podczas wykonywania projektu wspierałem się nastepującymi stronami:

1. Introsort Wikipedia
2. Mergesort Wikipedia
3. Quicksort Wikipedia
4. Sortowania Wikipedia
5. Geeksforgeeks Merge sort.
6. Geeksforgeeks Quick sort
7. Geeksforgeeks Intro sort
8. Geeksforgeeks Insertion sort