Sprawozdanie: Zadania 6, 7, 8 - sortowanie

Mateusz Gujda

Czasy wykonywania algorytmów w zależności od **rozmiaru nieposortowanej tablicy** wyznaczone przy pomocy średniej z 1000 pomiarów ze względu na niedokładność wyników uzyskanych przy użyciu funkcji **time.time()**:

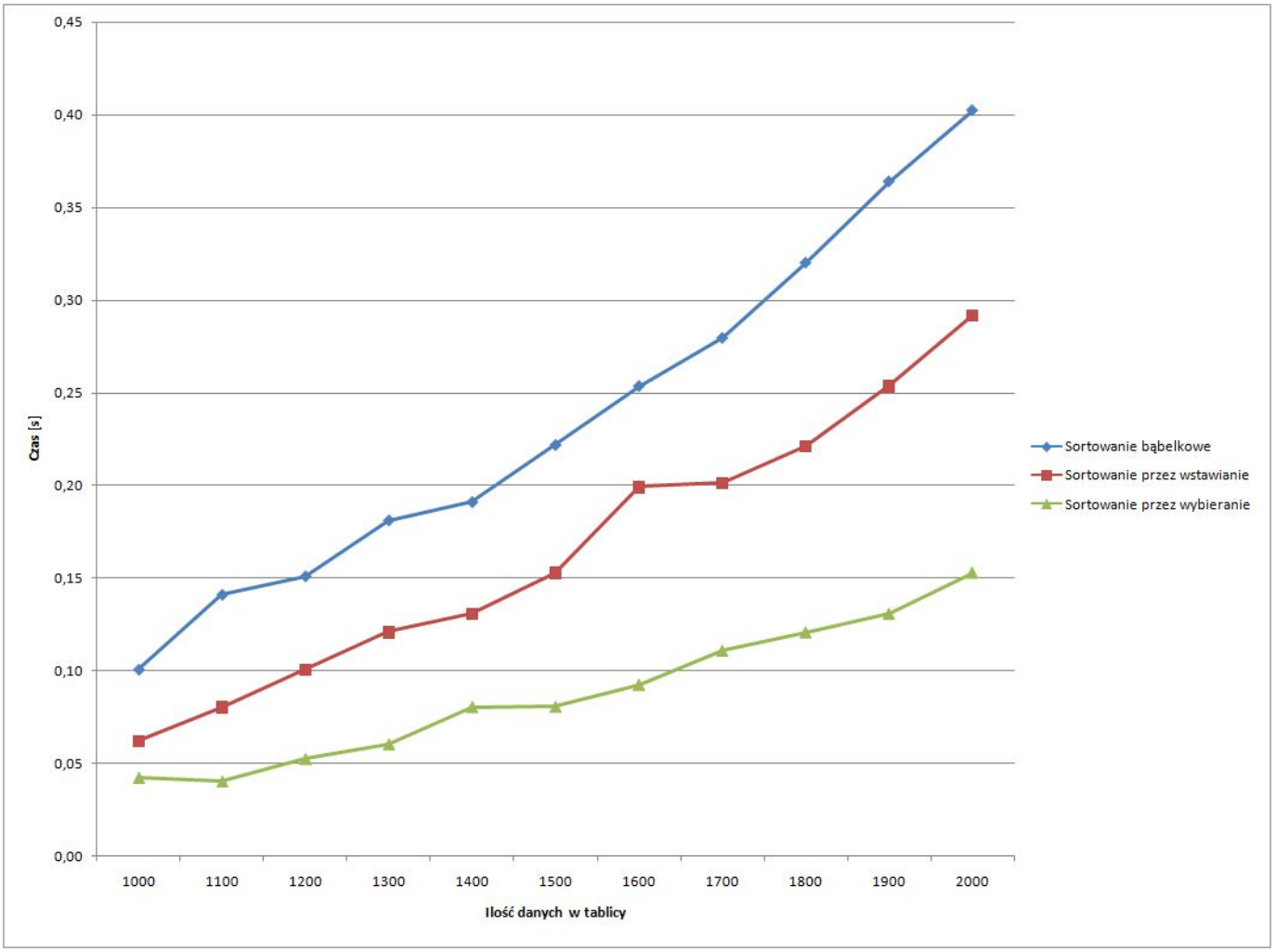
7 metod sortowania zostało podzielone na 2 grupy ze względu na swoją złożoność obliczeniową i uzyskanie wyniki czasowe.

Grupa 1:

Tabela zbiorcza zawierająca czasy działania algorytmów w sekundach

	Czas[s]		
Ilość danych	**Sortowanie bąbelkowe**	**Sortowanie przez wstawianie**	**Sortowanie przez wybieranie**
1000	0,100578546524047	0,062288999557495	0,042277812957764
1100	0,141025304794311	0,080428123474121	0,040284395217896
1200	0,150992870330810	0,100630283355712	0,052352428436279
1300	0,181199550628662	0,120974779129028	0,060361862182617
1400	0,191079139709472	0,130650520324707	0,080489397048950
1500	0,221879005432128	0,152862548828125	0,080528259277344
1600	0,253674745559692	0,199173450469970	0,092541456222534
1700	0,279670953750610	0,201313734054565	0,110851049423217
1800	0,320259571075439	0,221335411071777	0,120715856552124
1900	0,364163398742675	0,253681182861328	0,130816221237182
2000	0,402566194534301	0,291730165481567	0,152940034866333

Wykres przedstawiający ww. zależności:



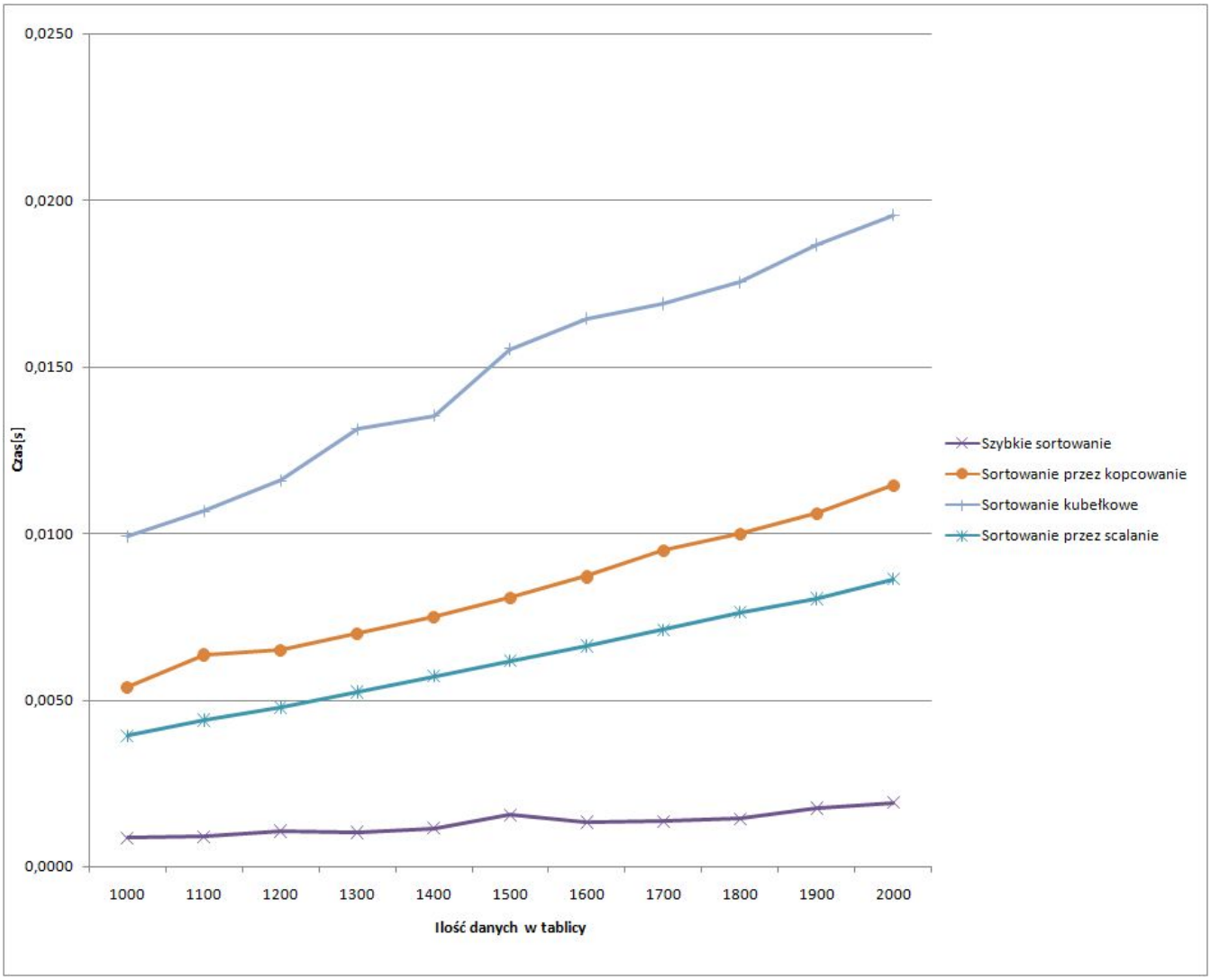
Powyższe algorytmy były testowane w tej grupie ze względu na swoją wyższą klasę złożoności obliczeniowej (zwykle n^2 w porównaniu do $n \log n$ algorytmów z grupy 2). Widać jak szybko przy niewielkim wzroście liczby danych wzrasta czas obliczeń.

Grupa 2:

Tabela zbiorcza zawierająca czasy działania algorytmów w sekundach

	Czas[s]			
Ilość danych	Szybkie sortowanie	Sortowanie przez scalanie	Sortowanie przez kopcowanie	Sortowanie kubełkowe
1000	0,000861665010452	0,003929591417313	0,005392764568329	0,009905514955521
1100	0,000885844707489	0,004401664972305	0,006358209371567	0,010655459403992
1200	0,001056529521942	0,004787209272385	0,006492878675461	0,011577265977859
1300	0,001028491258621	0,005243739128113	0,006994546175003	0,013124026298523
1400	0,001149500370026	0,005709182500839	0,007483819961548	0,013530245542526
1500	0,001549800157547	0,006173934221268	0,008077525615692	0,015530095577240
1600	0,001335820913315	0,006619467735291	0,008702480792999	0,016443383693695
1700	0,001355755090714	0,007102433681488	0,009483670234680	0,016889370441437
1800	0,001430042505264	0,007623664379120	0,009983632564545	0,017529949903488
1900	0,001741199254990	0,008028094053268	0,010608587741852	0,018654865741730
2000	0,001911910295486	0,008611626148224	0,011436647415161	0,019545422554016

Wykres przedstawiający ww. Zależności:



Powyższe zestawienie pokazuje jak istotne jest dbanie o optymalizację wykorzystywanych algorytmów. Zejście o klasę złożoności niżej umożliwia nam wykonywanie znacznie szybszych operacji na tych samych zbiorach danych.