

<b>Uniwersytet Zielonogórski</b>	Wykonali:	Grupa:	Nr ćwiczenia:	Ocena:
<b>Laboratorium</b>				
Temat ćwiczenia: Sortowanie wewnętrzne		Prowadzący:	Data wyk. ćw.	Data oddania

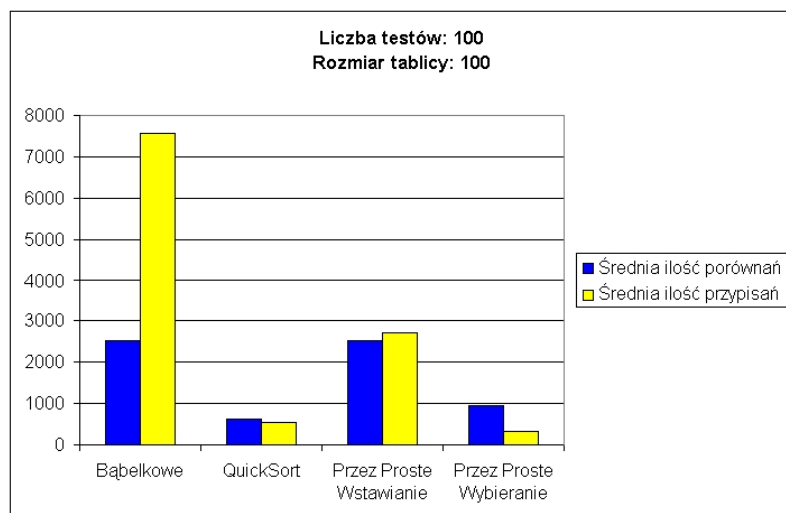
Na ostatnim laboratorium zajęliśmy się sortowaniem wewnętrznym. Z programu przedstawiającego różne metody sortowania wewnętrznego, otrzymaliśmy wyniki średniej ilości porównań i przypisań dla poszczególnych rodzajów sortowań: dla sortowania bąbelkowego, QuickSort, przez proste wstawianie i przez proste wybieranie.

Dla każdego sortowania przeprowadziliśmy badanie polegające na uporządkowaniu kolejno: 100, 500, 1000 i 5000 elementowej tablicy liczb. Liczba testów była stała i wynosiła 100.

Poniższa tabelka i wykres słupkowy ukazują wyniki badań. :

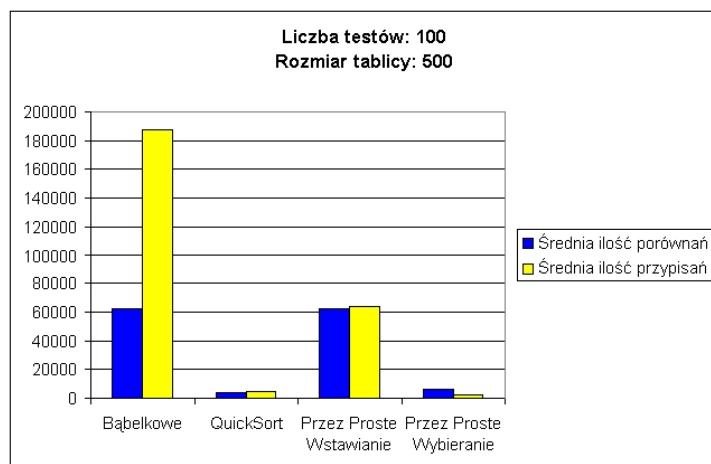
Rozmiar tablicy: 100:

	Bąbelkowe	QuickSort	Przez proste wstawianie	Przez proste wybieranie
Średnia ilość porównań	2519	636	2519	958
Średnia ilość przypisań	7557	551	2719	328



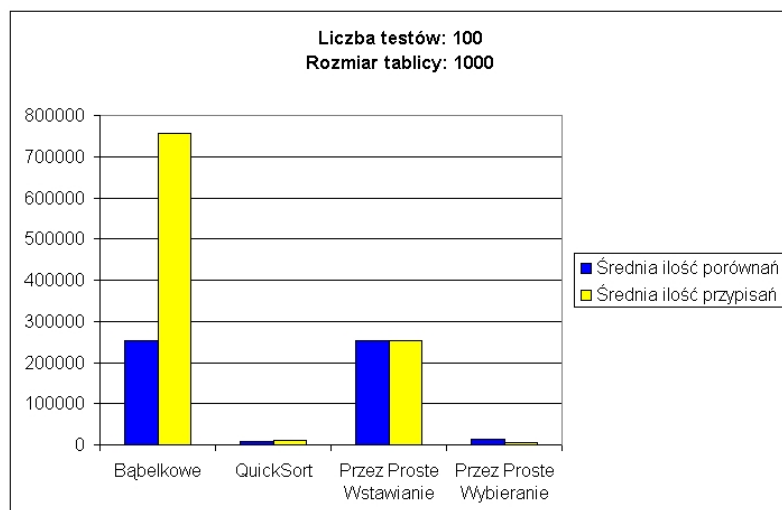
Rozmiar tablicy: 500:

	Bąbelkowe	QuickSort	Przez proste wstawianie	Przez proste wybieranie
Średnia ilość porównań	62584	4564	62584	6351
Średnia ilość przypisań	187753	3563	63592	2418



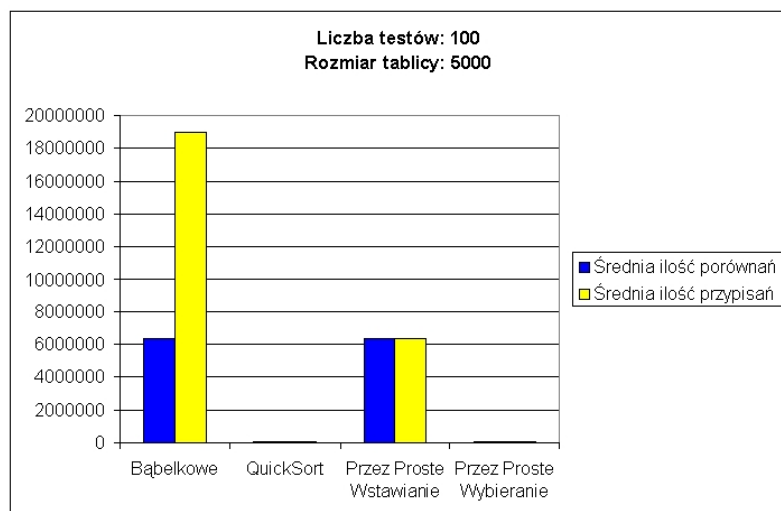
Rozmiar tablicy: 1000:

	Bąbelkowe	QuickSort	Przez proste wstawianie	Przez proste wybieranie
Średnia ilość porównań	251814	10371	251814	14088
Średnia ilość przypisań	755442	7849	253832	5529



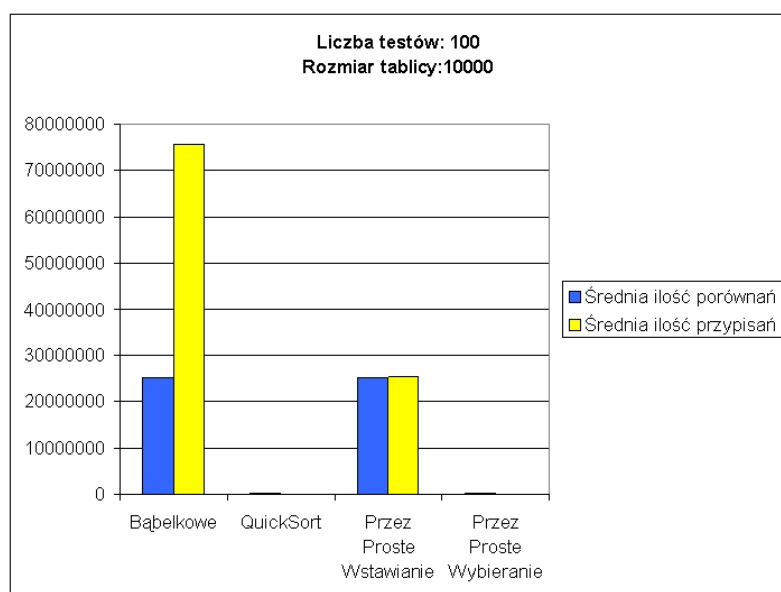
Rozmiar tablicy: 5000:

	Bąbelkowe	QuickSort	Przez proste wstawianie	Przez proste wybieranie
Średnia ilość porównań	6315329	65105	6315329	86976
Średnia ilość przypisań	18945986	47335	6325427	35913



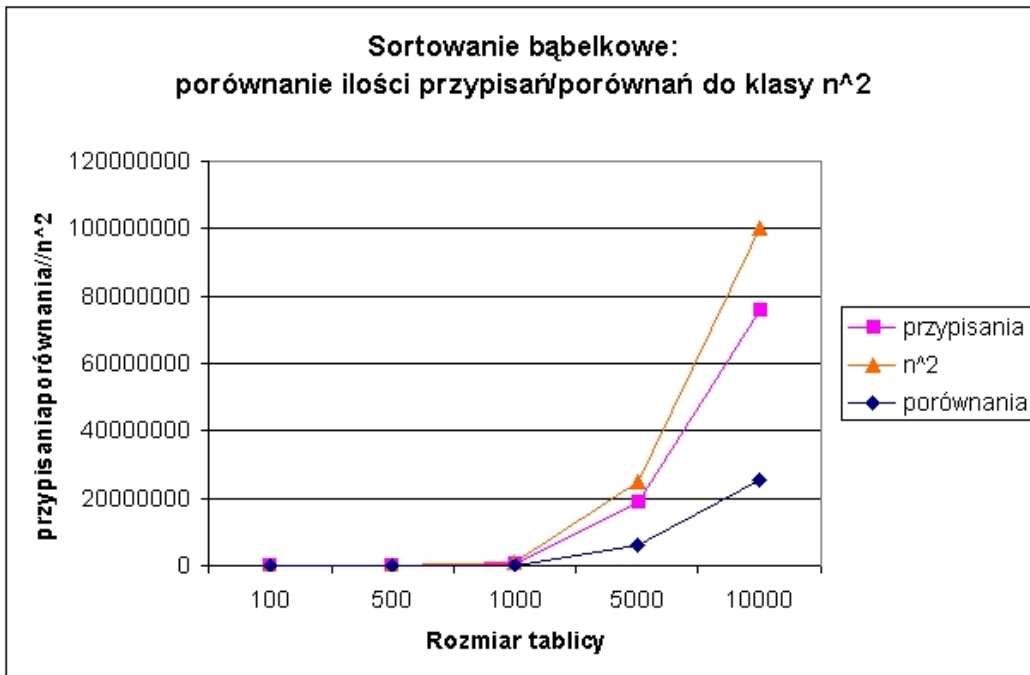
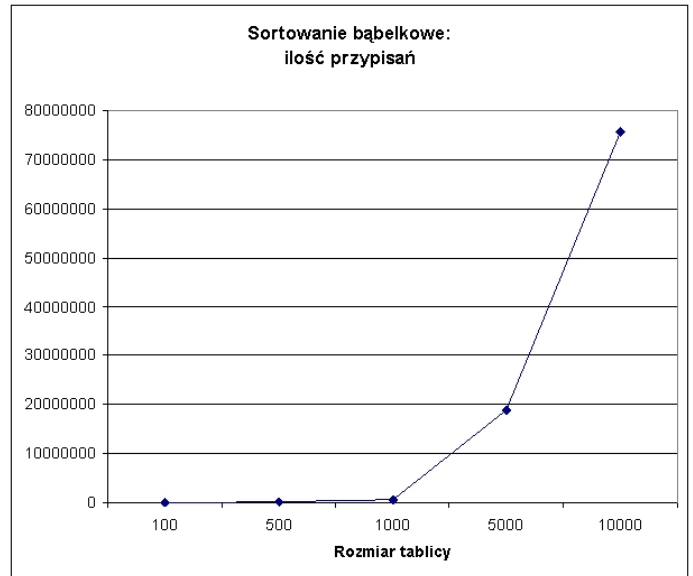
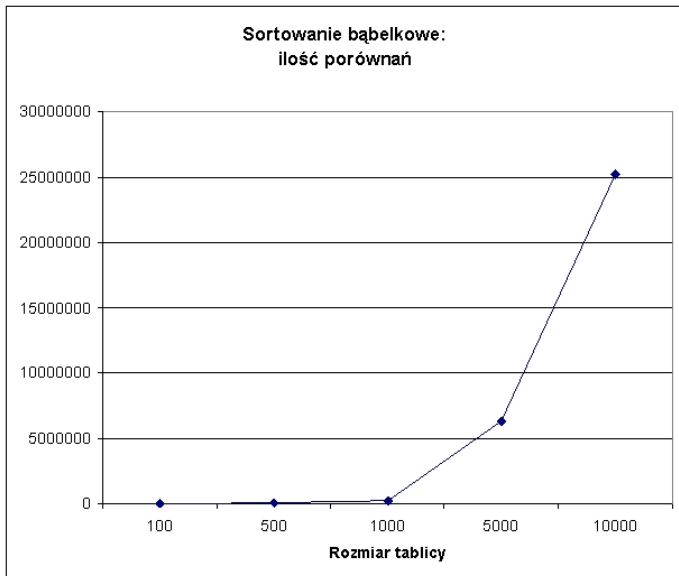
Rozmiar tablicy: 10000:

	Bąbelkowe	QuickSort	Przez proste wstawianie	Przez proste wybieranie
Średnia ilość porównań	25254289	142541	25254289	187578
Średnia ilość przypisań	75762867	101532	25274487	78639

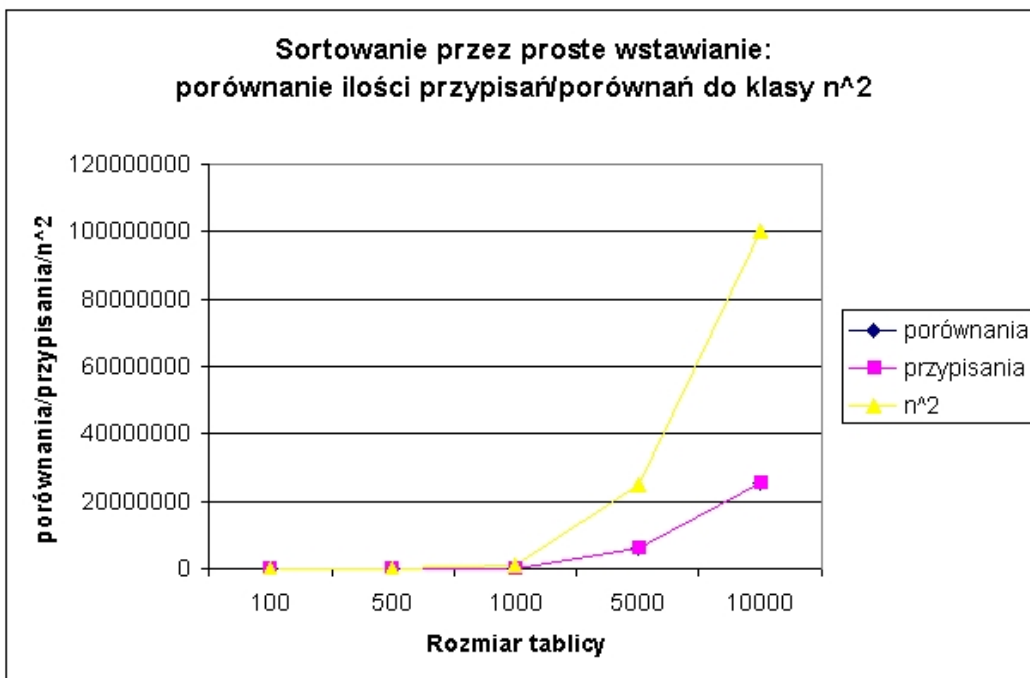
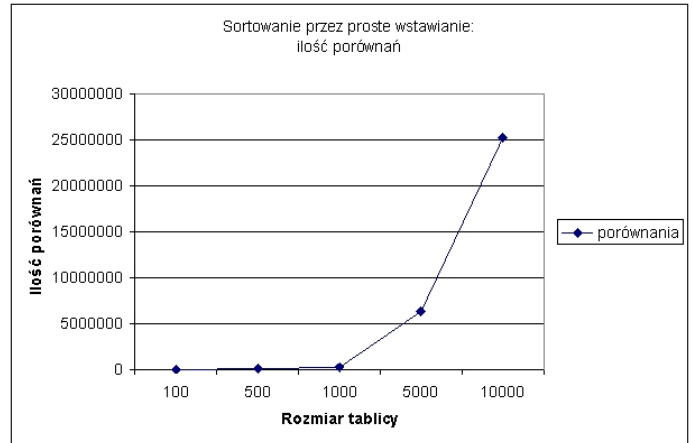
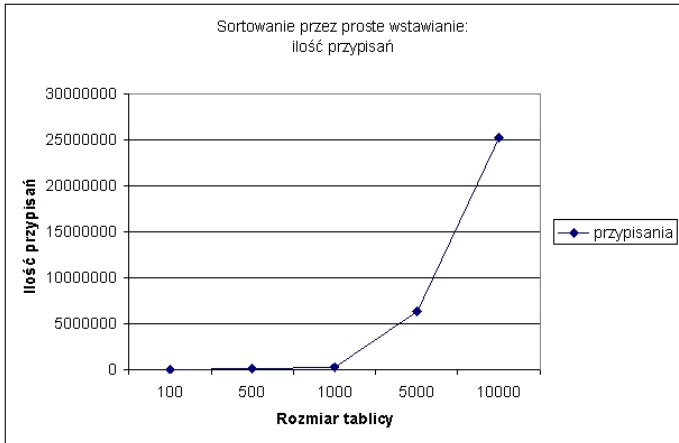


Jak widać na powyższych wykresach, sortowanie bąbelkowe i przez proste wstawianie potrzebują bardzo dużo porównań i przypisań. Są one klasy  $O(n^2)$ , co widać poniżej:

### Sortowanie bąbelkowe:

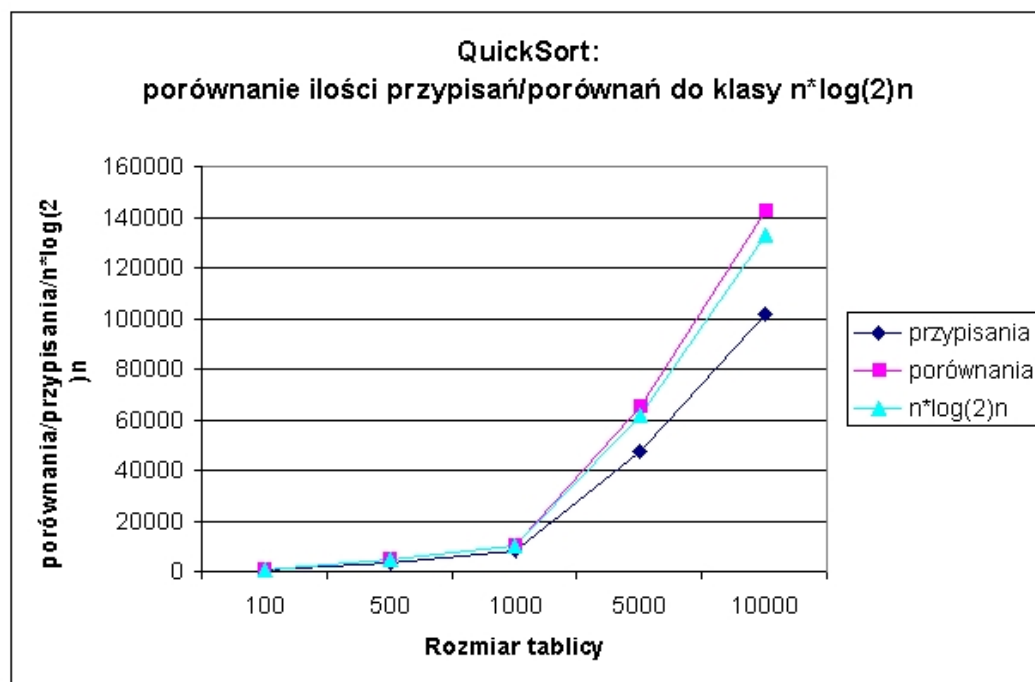
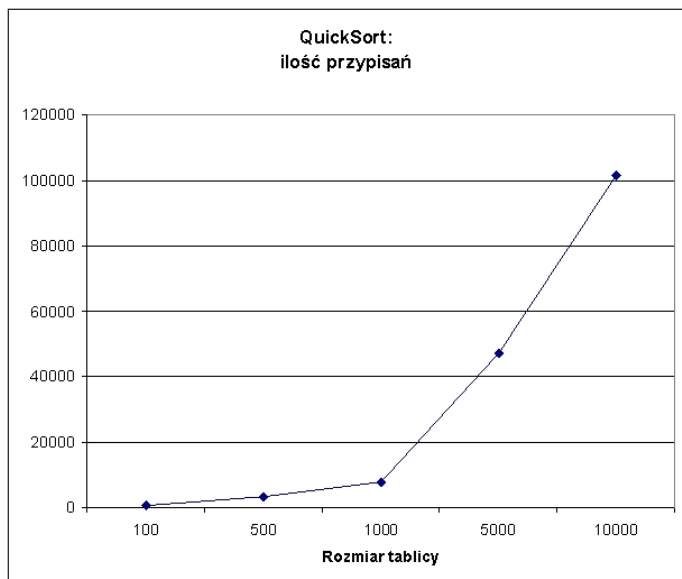
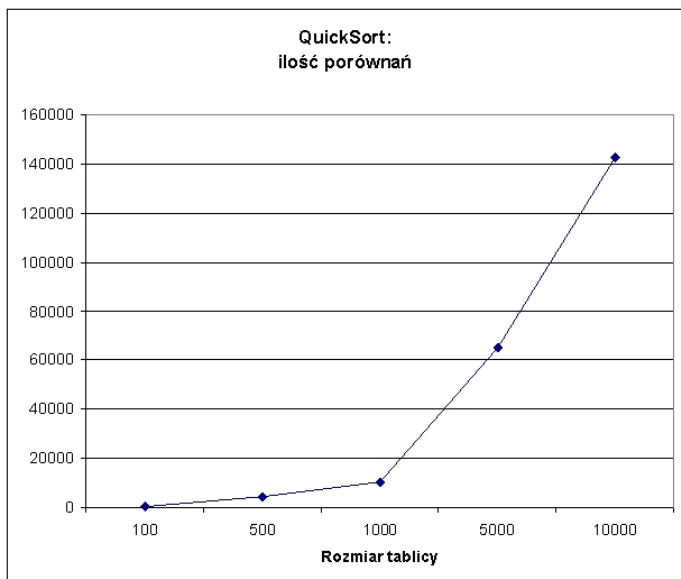


### Sortowanie przez proste wstawianie:

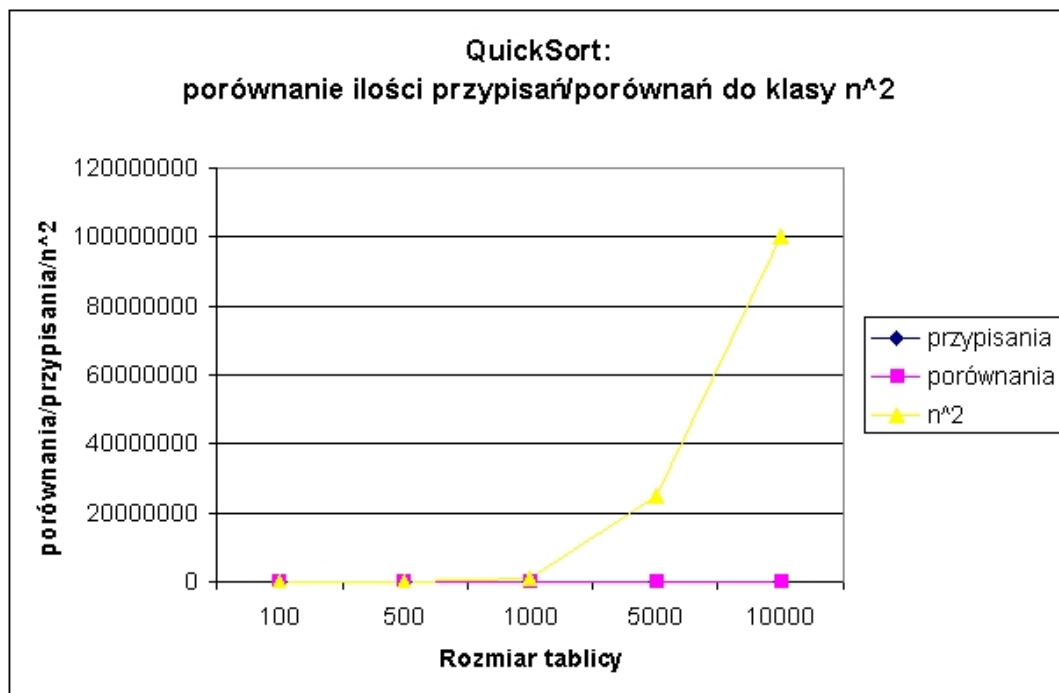


(przypisania i porównania pokrywają się)

Z kolei QuickSort jest już klasy  $N \cdot \log_2 N$ . Widać to wyraźnie na poniższych wykresach:

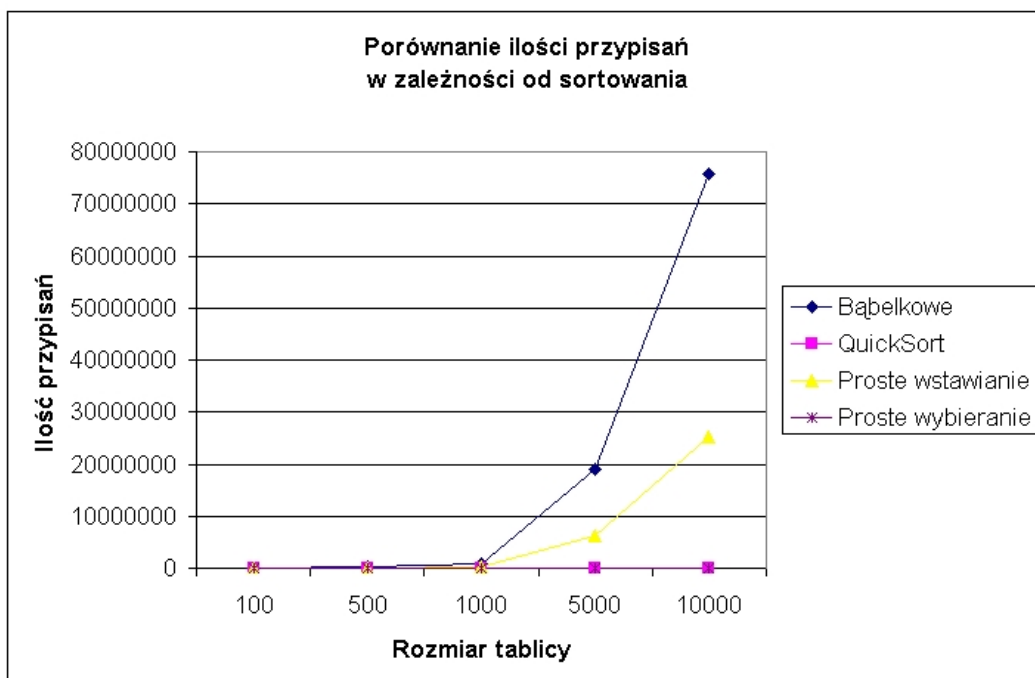


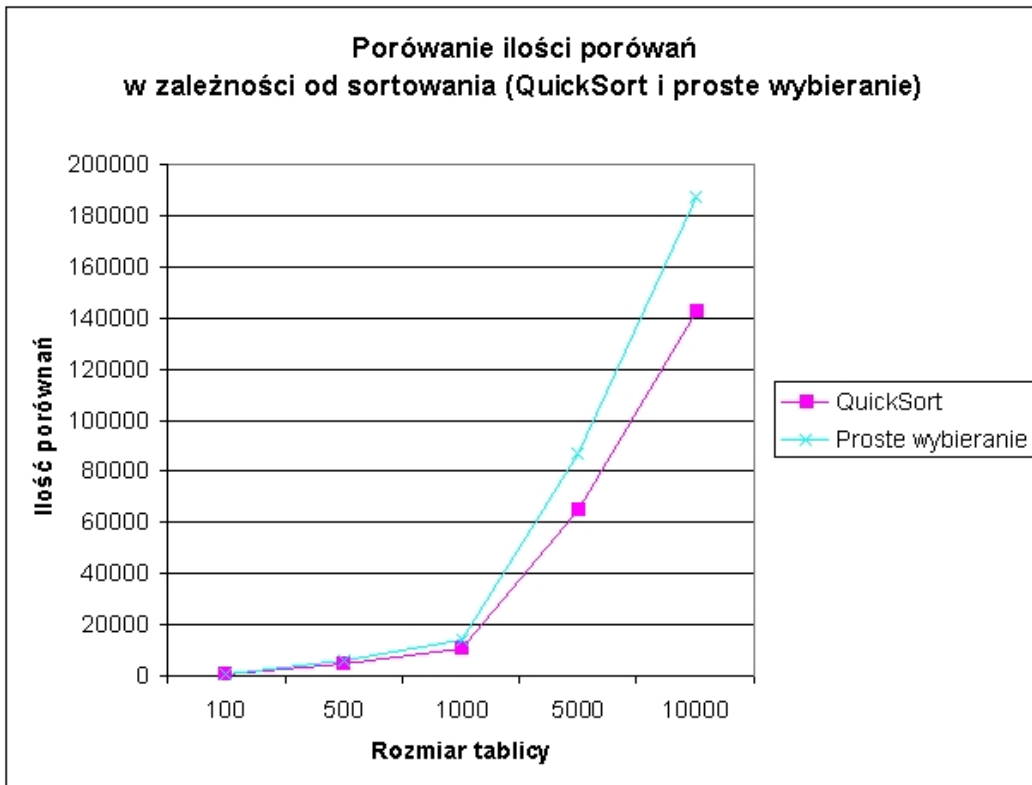
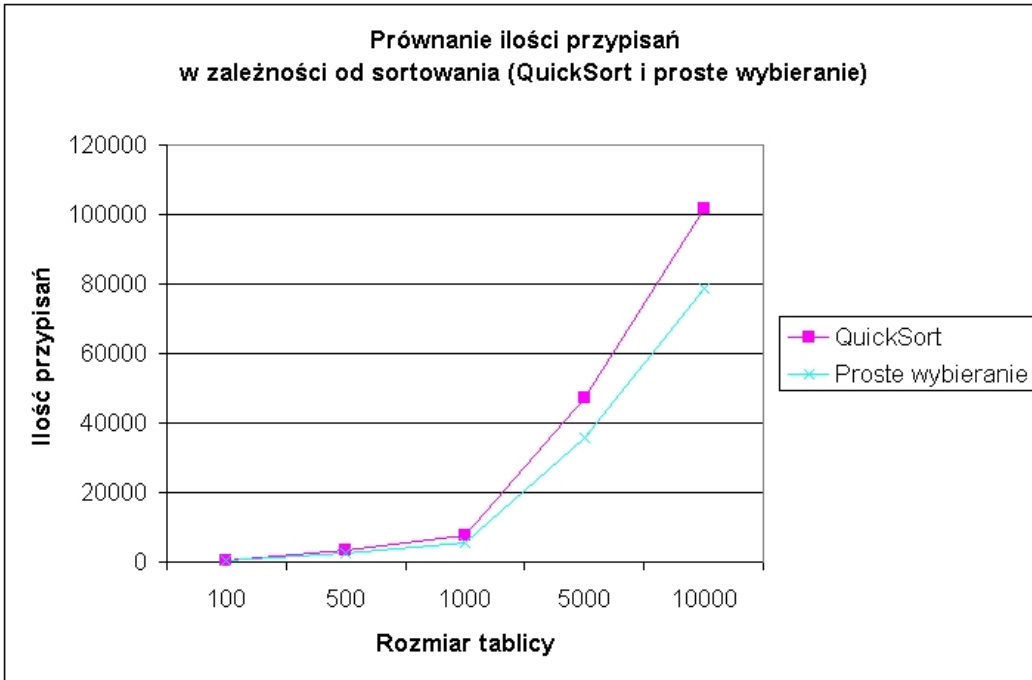
A to pokazanie kolosalnej różnicy między QuickSort a sortowaniem klasy  $n^2$ :

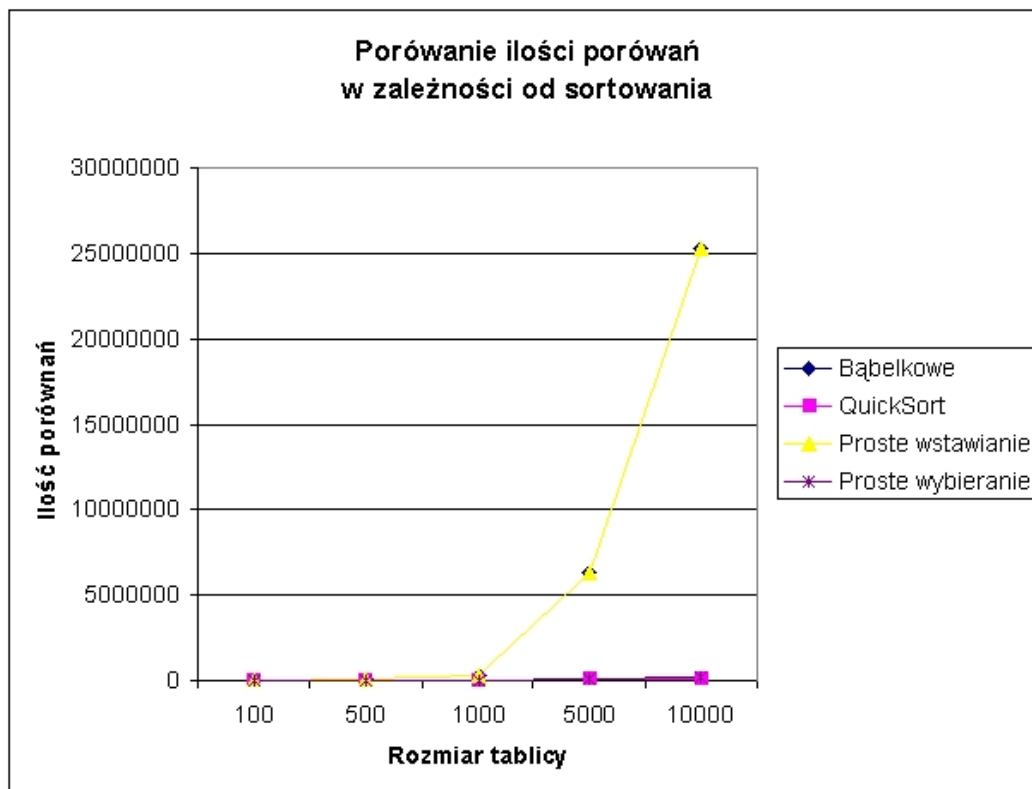


Porównania i przypisania dosłownie pokrywają się z osią X.

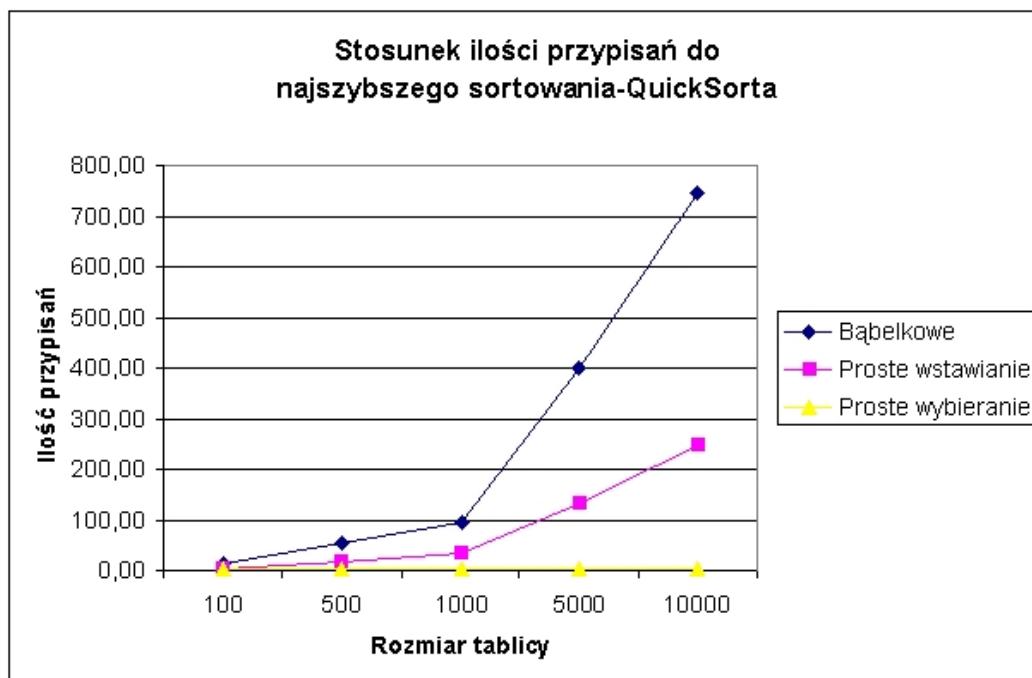
Poniższe wykresy porównują ilości porównań i przypisań w zależności od sortowania:

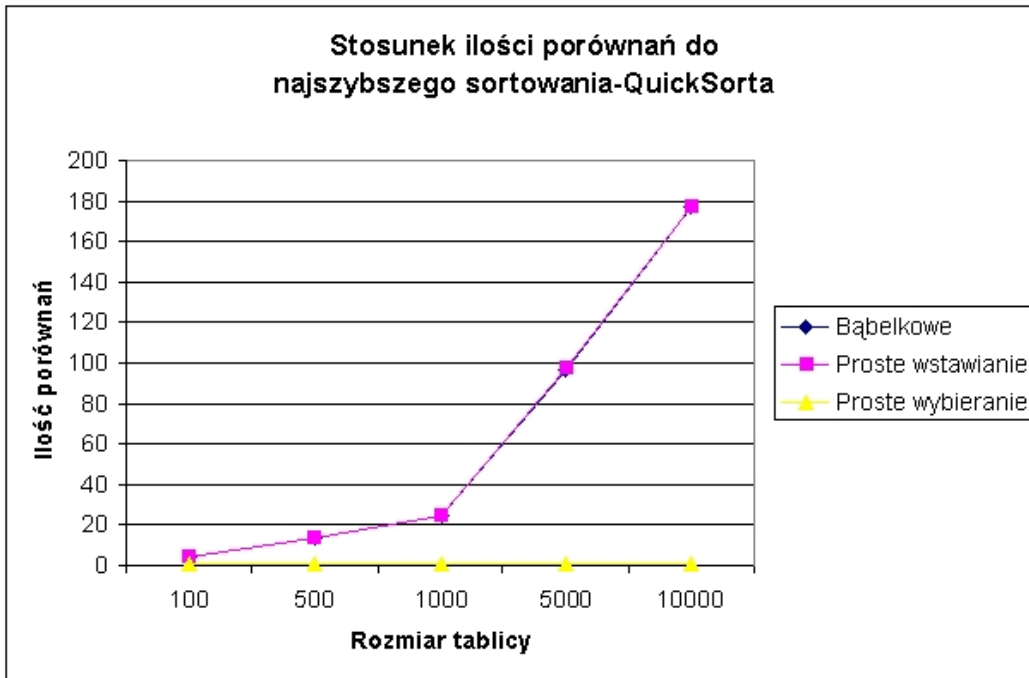




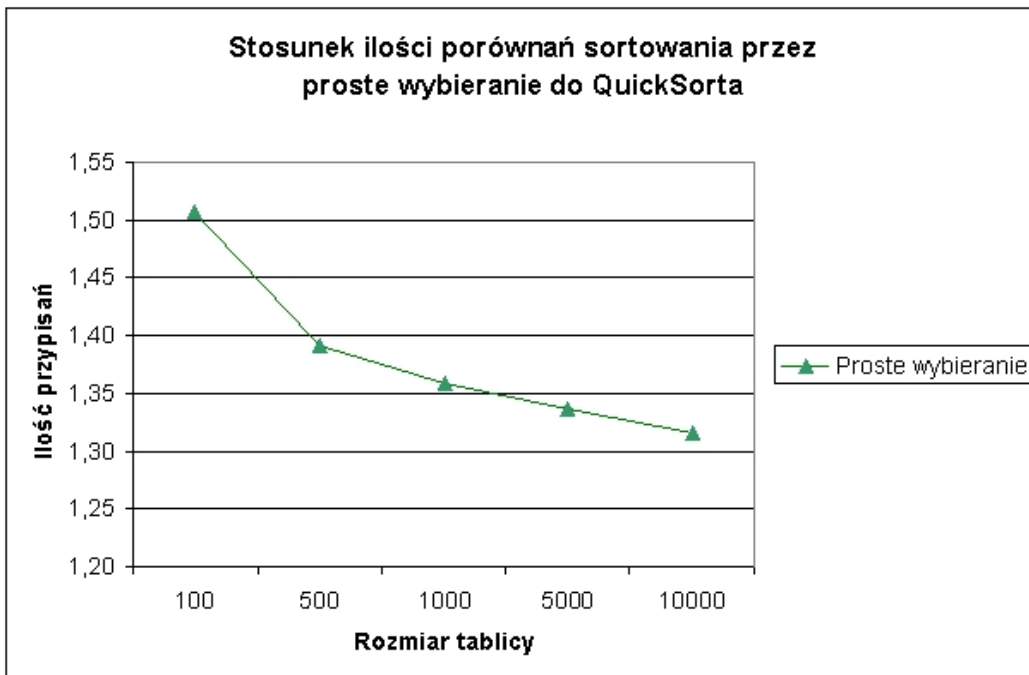


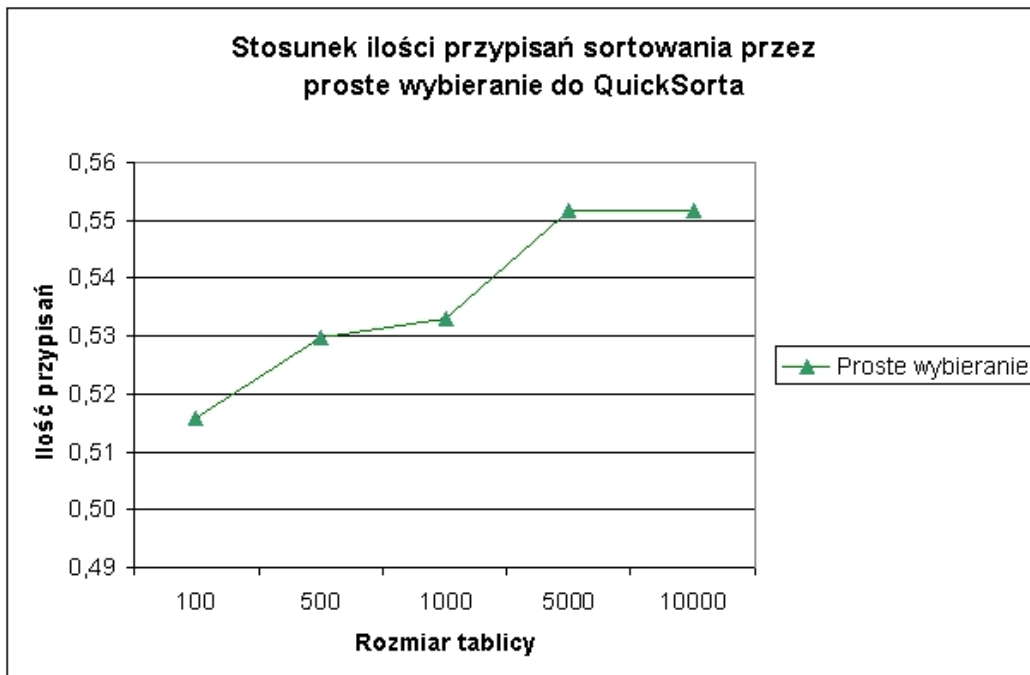
Okazało się, że najszybszym sortowaniem jest QuickSort. Porównaliśmy je do pozostałych sortowań. Za jednostkę obraliśmy wynik QuickSorta w poszczególnych rozmiarach tablic. Wykresy pokazują ile razy więcej nastąpiło porównań i przypisań w stosunku do QS:





Po dokładnym zbadaniu natrafiliśmy na ciekawą rzecz:





Okazało się, że ilość przypisań w sortowaniu przez proste wybieranie jest mniejsza prawie dwukrotnie od QuickSorta. Ilość porównań w sortowaniu przez proste wybieranie jest z kolei o 50% większa od QS, ale spada przy większej ilości danych i poza tym nie tak duża jak w pozostałych sortowaniach.

## Wnioski:

Sortowanie bąbelkowe okazało się nawolniejszym sortowaniem. Jest złożoności  $O(n^2)$ . Podczas wykonywania cechował się bardzo dużą ilością przypisań i porównań. Z drugiej strony jest bardzo prosty w implementacji i niezależnie od danych następuje identyczna ilość porównań. Algorytm ten nie nadaje się do sortowania dużej ilości danych.

Sortowanie przez proste wstawianie choć też jest złożoności  $O(n^2)$  i występuje w nim duża liczba porównań, to jest szybsze oraz bardziej efektywne od sortowania bąbelkowego. Dla danych posortowanych brak przestawień elementów. Z kolei dla danych posortowanych odwrotnie następuje najwięcej przestawień. Algorytm ten jest prosty w implementacji, przez co może być wykorzystywany zamiast sortowania bąbelkowego.

Sortowanie przez proste wybieranie ma takie same zalety i wady, co przez proste wstawianie. Jest złożoności  $O(n^2)$ , występuje w nim duża ilość porównań. Ilość przestawień wynosi  $n-1$ , liczba porównań natomiast jest proporcjonalna do  $n^2$ . Cech te predestynują sortowanie przez proste wybieranie do obróbki zbiorów z dużymi polami danych i małymi kluczami, gdyż w takich zastosowaniach koszt przemieszczenia danych dominuje nad kosztem porównań-co widoczne było na w/w wykresach-a żaden inny algorytm nie dokonuje mniejszej ilości przestawień podczas sortowania.

Ostatni przedstawiony algorytm sortowania, QuickSort, jest bardzo trudny w implementacji. Jest złożoności  $O(n \cdot \log_2 n)$ , ale dla niekorzystnych danych złożoność może nawet spaść do  $O(n^2)$ .

Rekurencyjna wersja tego algorytmu posiada ograniczenia wielkości stosu. W najlepszym przypadku liczba zmian wynosi  $n/6$ , a w najgorszym  $n^2/6$ . Jednak mimo tych wad, algorytm ten posiada jedną niezaprzeczalną zaletę. Jest najszybszym z w/w sortowań. Przy małych zbiorach danych nie odczuwamy większej różnicy przy wykorzystywaniu poszczególnych algorytmów, jednak przy większej ilości danych zdecydowanie wybór pada na algorytm QuickSort (co ukazaliśmy na w/w wykresach).