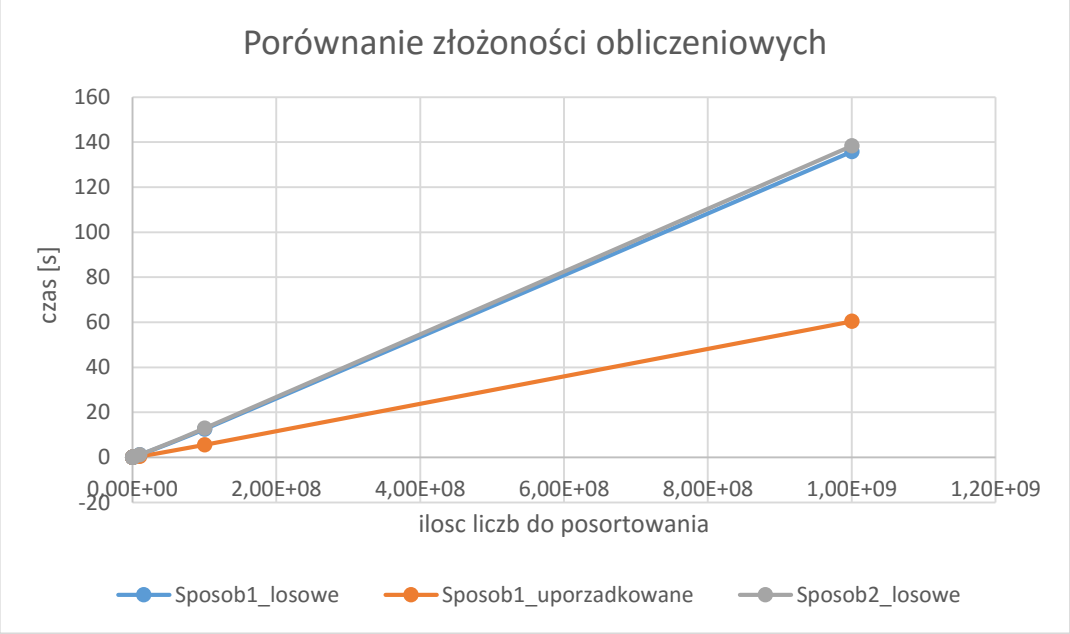
Jakub Jarek 218490 termin: poniedziałek 8:15

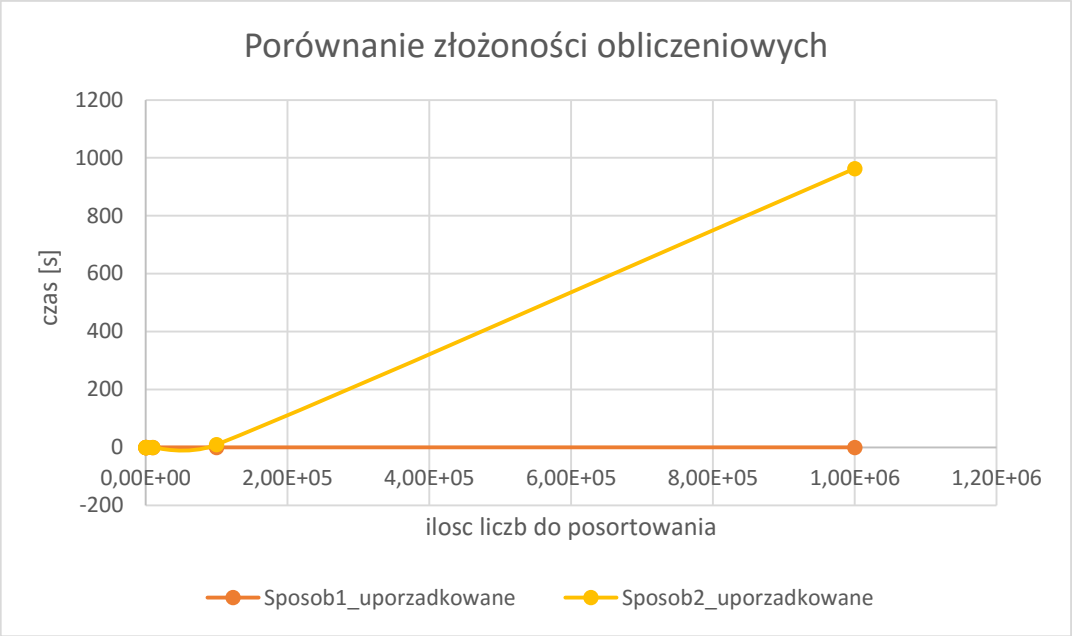
1. Sortowanie szybkie

Dobór pivotowania:

- Sposób 1 – za oś wybieramy wartość z środkowym numerem indeksu $\frac{lewy\ indeks\ graniczny+prawy\ indeks\ graniczny}{2}$
- Sposób 2 – za oś wybieramy lewą granicę sortowania (lewy indeks graniczny)

Ilość liczb do posortowania	Sposób 1		Sposób 2	
	losowe	uporządkowane malejąco	losowe	uporządkowane malejąco
	średni	optymistyczny	średni	pesymistyczny
10^1	0,000001	0,000001	0,000002	0,000001
10^2	0,000008	0,000004	0,000032	0,000014
10^3	0,000103	0,00005	0,000441	0,001001
10^4	0,001006	0,000378	0,003253	0,096768
10^5	0,017593	0,004858	0,013483	9,57846
10^6	0,107486	0,040849	0,109012	962,969
10^7	1,16473	0,485863	1,17327	
10^8	12,5051	5,58724	12,9221	
10^9	135,762	60,3756	138,295	





Wnioski:

Przez rzadkie pomiary przy dużym zakresie liczb nie jesteśmy w stanie określić złożoności obliczeniowej na podstawie kształtu wykresu. Biorąc jednak pod uwagę wartości z tabeli możemy słusznie podejrzewać, iż algorytm ma złożoność obliczeniową O(nlog(n)) dla dowolnego zestawu liczb (pierwszy sposób pivotowania) oraz dla zestawu losowych liczb (drugi sposób pivotowania).

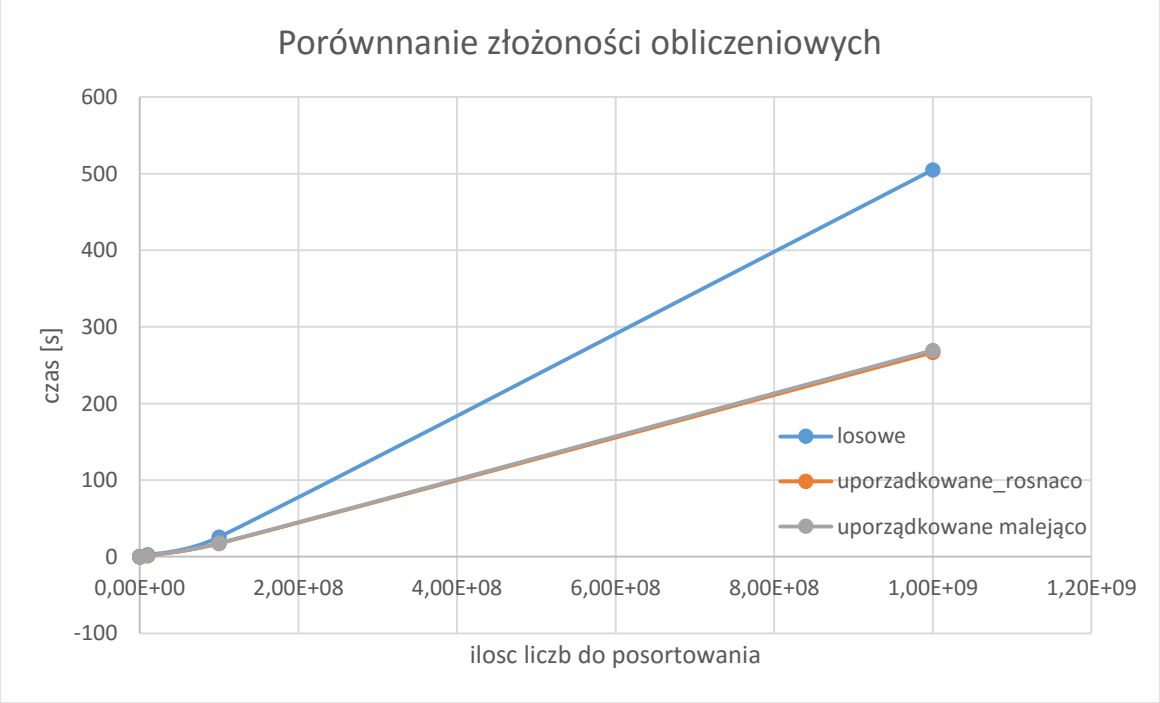
Uporządkowany (posortowany malejąco) zbiór liczb dla pierwszego sposobu pivotowania okazał się przypadkiem optymistycznym.

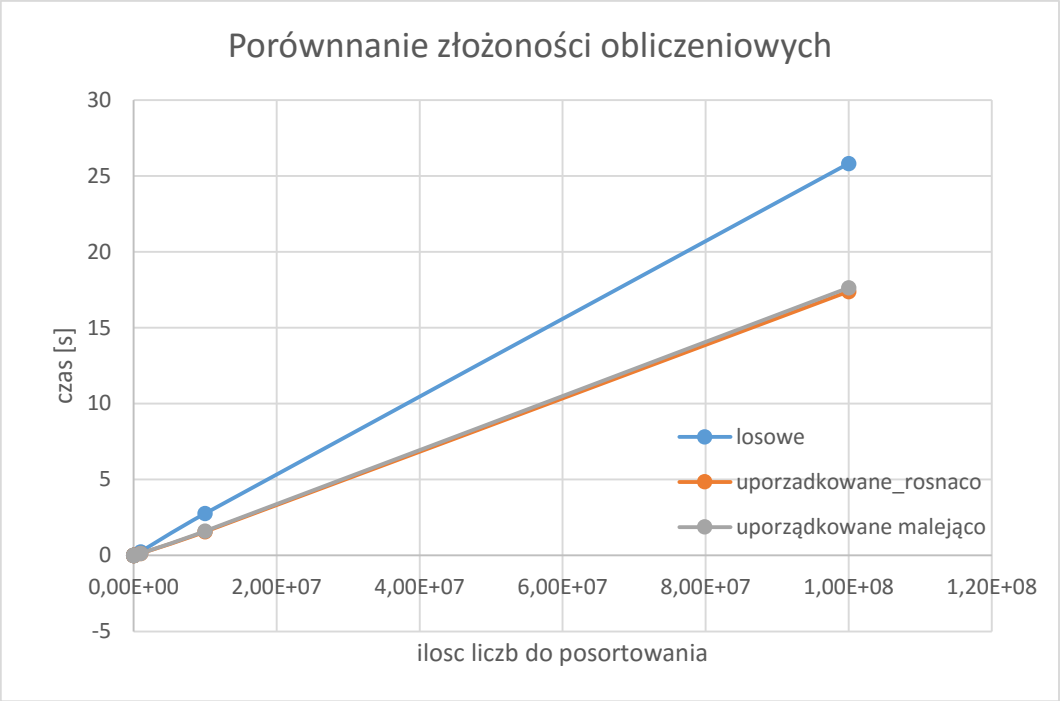
Uporządkowany (posortowany malejąco) zbiór liczb dla drugiego sposobu pivotowania okazał się przypadkiem pesymistycznym – osiągnął złożoność $O(n^2)$. Jest to spowodowane tym, iż wybierając oś jako lewy skrajny element posortowanej podtablicy zawsze trafiamy na wartość minimalną lub maksymalną tejże podtablicy. Sortowanie szybkie staje się wtedy nieefektywne, ponieważ taka podtablica nie jest praktycznie dzielona, mamy jedynie o jeden element (osiowy) mniej, a pozostałą część dzielimy dalej.

Wybór osi dla przypadku średniego (zestaw liczb losowych) jest dowolny. Wykres złożoności dla obu przypadków praktycznie się pokrywa.

2. Sortowanie przez scalanie

Ilość liczb do posortowania	losowo	uporządkowane rosnąco	uporządkowane malejąco
	średni	optymistyczny	optymistyczny
10^1	0,000001	0,000001	0,000001
10^2	0,000013	0,000007	0,000009
10^3	0,000161	0,000078	0,000102
10^4	0,002266	0,001017	0,001247
10^5	0,022835	0,011879	0,016653
10^6	0,249713	0,13579	0,139518
10^7	2,76256	1,56381	1,60482
10^8	25,8168	17,3851	17,6306
10^9	504,954	266,649	269,435





Wnioski:

Ponownie przez rzadkie pomiary przy dużym zakresie liczb nie jesteśmy w stanie określić złożoności obliczeniowej na podstawie kształtu wykresu.

Pojawia się również bardzo niepokojąca sytuacja w kwestii testowanego algorytmu Mergesort. Według założeń teoretycznych algorytm powinien pracować dla dowolnych danych wejściowych ze złożonością O(nlog(n)). Mój algorytm odbiega od tego założenia dla dużych zestawów danych wejściowych $>10^6$. O ile na II wykresie (zakres 10^8) możemy próbować przeprowadzić wykres logarytmu, o tyle na I wykresie (zakres 10^9) już na pewno nie jesteśmy w stanie tego zrobić.

Próbą wytłumaczenia naszej niezgodności może być fakt, iż Merge sort ma większą złożoność pamięciową niż Quicksort, co dla dużych zbiorów liczb może oznaczać wykończenie zasobów pamięci RAM, co znacznie spowalnia cały proces wykonywania algorytmu.