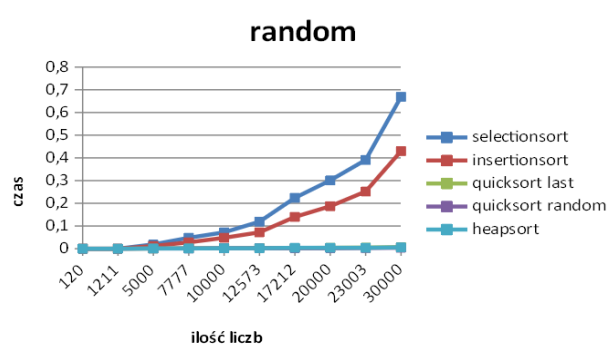
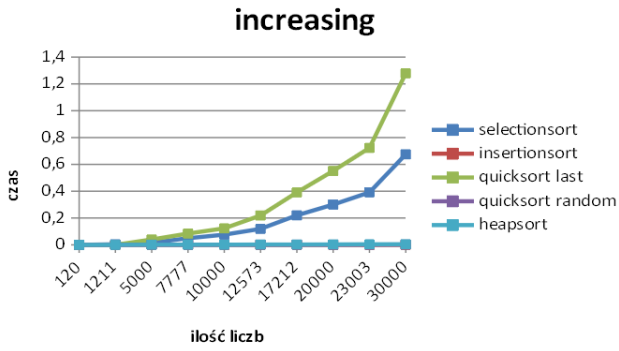
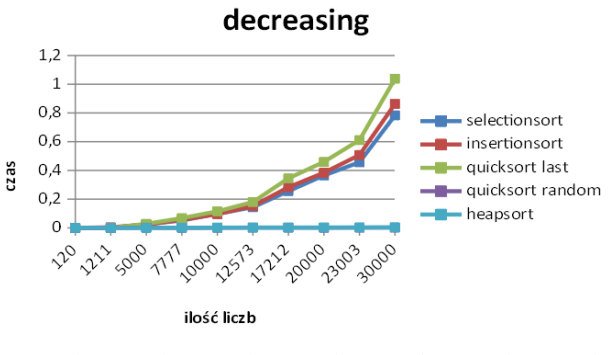
Algorytmy sortowania - Sprawozdanie

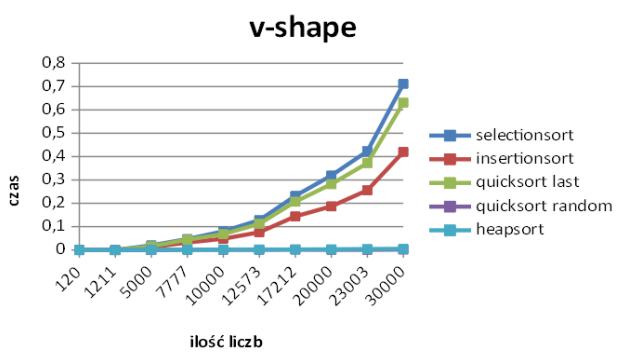
Igor Marchlewski 145174
Wojciech Warmuz 145157

Wykresy porównawcze dla różnych typów rozkładu danych









Selection Sort

	ilość liczb	random	increasing	decreasing	v-shape
selectionsort	120	0	0	0	0
	1211	0	0,004	0	0
	5000	0,019	0,016	0,024	0,02
	7777	0,048	0,049	0,052	0,047
	10000	0,072	0,076	0,097	0,08
	12573	0,119	0,119	0,144	0,128
	17212	0,224	0,22	0,256	0,232
	20000	0,301	0,3	0,365	0,319
	23003	0,391	0,391	0,457	0,423
	30000	0,669	0,675	0,783	0,712

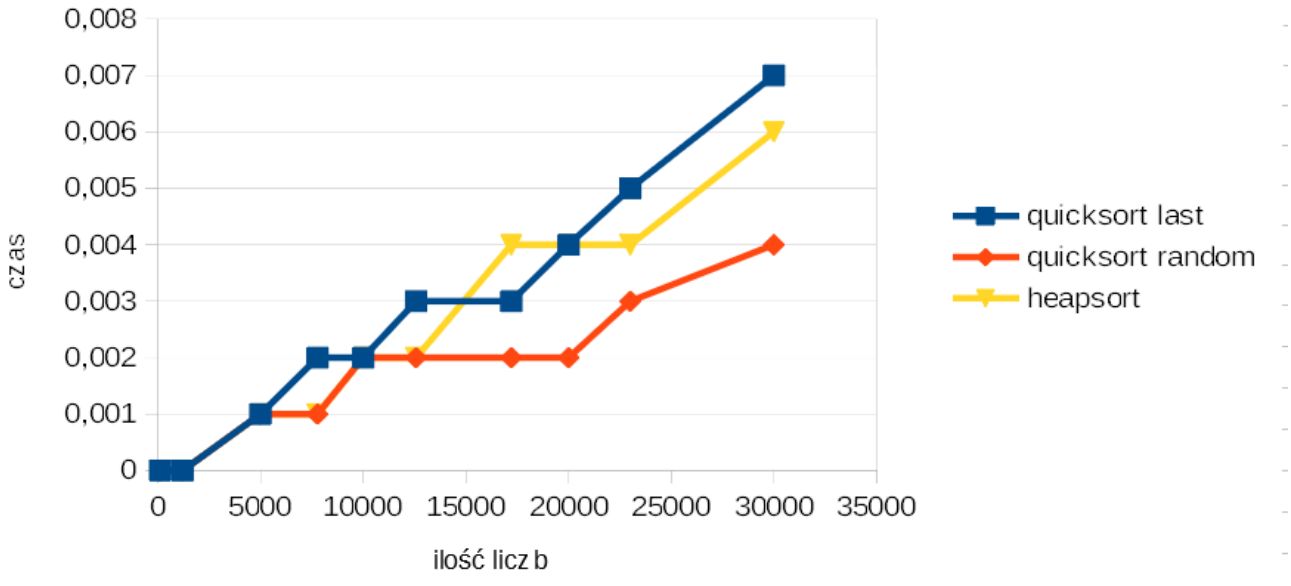
Selection sort zdecydowanie najwolniej poradził sobie z sortowaniem tablicy o losowym rozkładzie danych. Czas jego pracy nie jest zależny od rozkładu danych w tablicy więc nie da się zidentyfikować zarówno best jak i worst case scenario. Złożoność obliczeniową algorytmu selection sort można opisać jako O(n^2).

Insertion Sort

	ilość liczb	random	increasing	decreasing	v-shape
insertionsort	120	0	0	0	0
	1211	0	0	0,004	0
	5000	0,012	0	0,024	0,012
	7777	0,028	0	0,056	0,032
	10000	0,048	0	0,096	0,048
	12573	0,072	0	0,152	0,076
	17212	0,14	0	0,283	0,144
	20000	0,187	0	0,384	0,187
	23003	0,252	0	0,507	0,256
	30000	0,43	0	0,863	0,42

Insertion sort był nieznacznie szybszy od algorytmu selection sort podczas sortowania tabeli o losowym rozkładzie wartości. Najlepszy przypadek dla tego algorytmu zachodzi w momencie, gdy wartości w tabeli poddanej sortowaniu były już posortowane. W tej sytuacji złożoność obliczeniową algorytmu opisuje zależność liniowa O(n). Najgorszym przypadkiem danych wejściowych jest tabela posortowana malejąco, złożoność obliczeniowa opisywana jest wtedy jako O(n^2). Rozkład v-kształtny danych został posortowany w czasie podobnym do rozkładu losowego, ponieważ składa się po równo z najlepszego jak i najgorszego przypadku danych wejściowych.

Wykres porównawczy algorytmów średniej złożoności O(n*log n)



dla rozkładu losowego

Quick sort

	ilość liczb	random	increasing	decreasing	v-shape
quicksort last	120	0	0	0	0
	1211	0	0	0	0
	5000	0,001	0,04	0,028	0,016
	7777	0,002	0,084	0,068	0,044
	10000	0,002	0,123	0,115	0,068
	12573	0,003	0,219	0,18	0,112
	17212	0,003	0,391	0,345	0,207
	20000	0,004	0,551	0,459	0,282
	23003	0,005	0,723	0,611	0,372
	30000	0,007	1,279	1,038	0,631

	ilość liczb	random	increasing	decreasing	v-shape
quicksort random	120	0	0	0	0
	1211	0	0	0	0
	5000	0,001	0,001	0,001	0
	7777	0,001	0,001	0	0,001
	10000	0,002	0,001	0,001	0,001
	12573	0,002	0,001	0,002	0,001
	17212	0,002	0,001	0,002	0,002
	20000	0,002	0,001	0,001	0,001
	23003	0,003	0,001	0,002	0,001
	30000	0,004	0,002	0,002	0,002

Algorytm quick sort był znacznie szybszy w posortowaniu tabeli o losowym rozkładzie od algorytmów selection sort i insertion sort oraz nieznacznie szybszy (w wariancie w którym elementem rozdzielającym był ostatni element) od heap sorta.

Rozważane są dwie wariacje algorytmu quick sort. W pierwszym z nich element rozdzielający jest zawsze ostatnim elementem tablicy, natomiast w drugim element rozdzielający jest wybierany losowo dla każdej tablicy i jej fragmentów.

Quick sort, którego elementem rozdzielającym był ostatni element tablicy, nie poradził sobie dobrze z preposortowanymi danymi. W najgorszym przypadku, gdy jako dane wejściowe dostaje tablice już posortowane, jego złożoność obliczeniowa jest opisywana przez O(n^2).

Najlepszym przypadkiem dla obu wariantów jest wybór mediany jako elementu rozdzielającego w każdym rekurencyjnym wywołaniu funkcji podziału. Złożoność obliczeniowa jest opisywana wtedy przez O(n*log n). Średnia złożoność obliczeniowa jest zbliżona do przypadku optymistycznego.

Heap sort

	ilość liczb	random	increasing	decreasing	v-shape
heapsort	120	0	0	0	0
	1211	0	0	0	0
	5000	0,001	0,001	0	0,001
	7777	0,001	0,001	0,001	0,001
	10000	0,002	0,001	0,002	0,002
	12573	0,002	0,002	0,002	0,002
	17212	0,004	0,002	0,002	0,002
	20000	0,004	0,003	0,003	0,003
	23003	0,004	0,004	0,003	0,004
	30000	0,006	0,004	0,004	0,005

Heap sort w sortowaniu liczb losowych był nieznacznie wolniejszy od algorytmu quick sort ze stałym elementem rozdzielającym. Unika jednak problemu najgorszego przypadku związanego z posortowanymi tablicami wejściowymi lub dodatkowych kroków związanych z wyborem losowego elementu rozdzielającego. Zarówno w najlepszym jaki i w najgorszym przypadku złożoność obliczeniowa algorytmu heap sort jest opisywana przez O(n*log).