SPRAWOZDANIE

AiSD

Metody Sortowania

Bartłomiej Kowalewski, 145204, i5.2
Jakub Koza, 145436, i5.2

Pomiary wykonane zostały na sprzęcie:

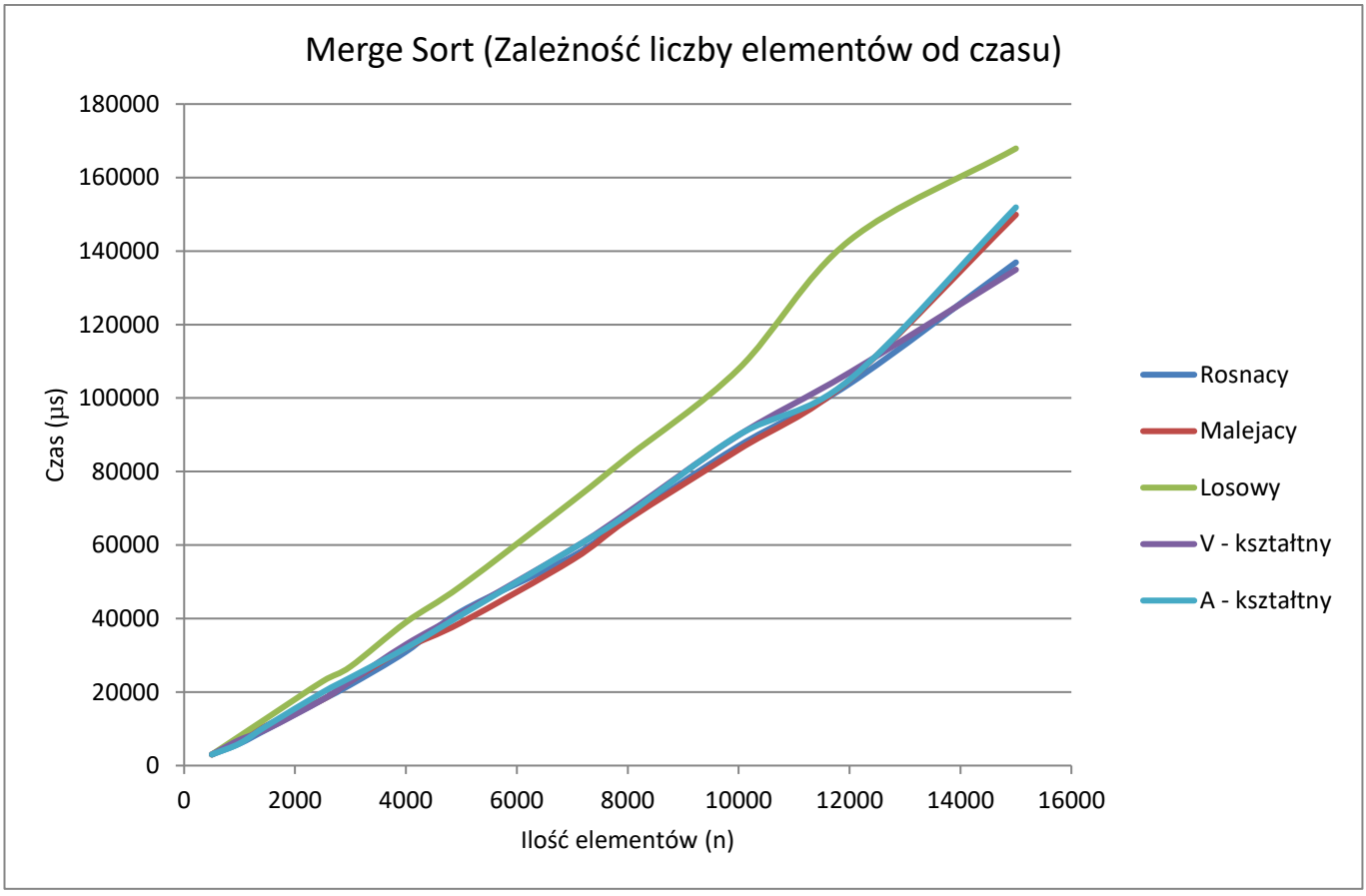
Laptop Lenovo G710
Windows 10 Home 64-bit
Procesor: Intel Core i7-4702MQ (2.2-3.2 GHz)
Pamięć RAM: 16 GB
Dysk twardy: 1TB SSHD
Karta graficzna: Intel HD Graphics 4600

W całości sprawozdania czas podawany jest w mikrosekundach.

1. Zależności czasów obliczeń (t) od liczby elementów tablicy (n) *(Pokazujące zależność algorytmów od danych wejściowych)*

   a) Merge Sort

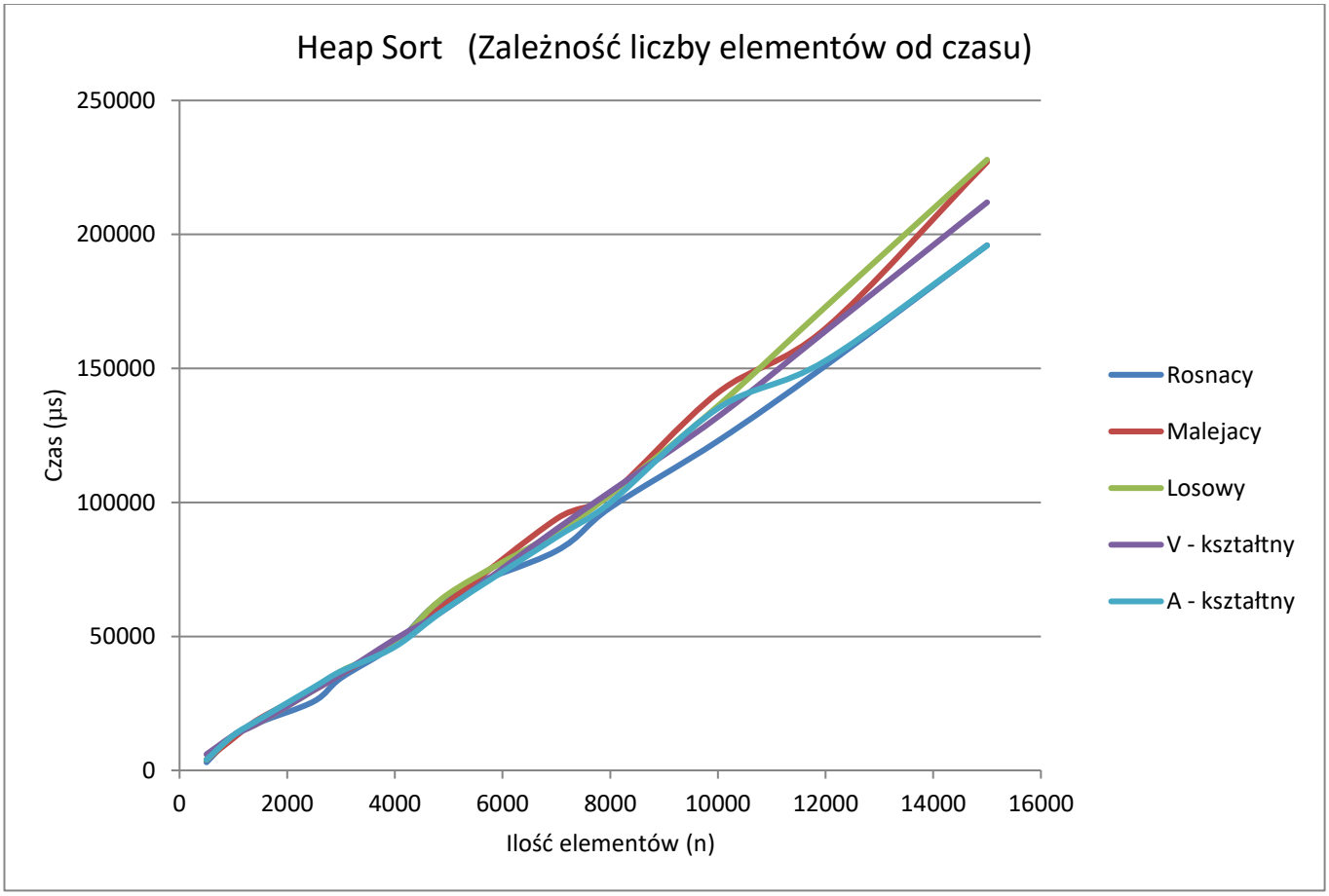
Merge Sort (Zależność liczby elementów od czasu)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	2999	2999	3011	3018	3018
1000	5981	7009	8015	6997	6016
1500	9951	11007	13022	9968	10934
2500	17973	17994	22941	17990	19963
3000	22004	22986	26941	22987	23963
4000	30983	31980	38995	32964	31995
5000	41975	38937	48930	41451	40986
7000	56924	55987	71941	58980	58983
8000	67246	66918	83970	68935	68359
10000	86960	85968	107951	89909	89944
12000	103892	104924	142902	106935	105014
15000	136921	149900	167924	134938	151897



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Merge Sort czasowo jest tak samo wydajny dla wszystkich pięciu rodzajów danych wejściowych.

b) Heap Sort

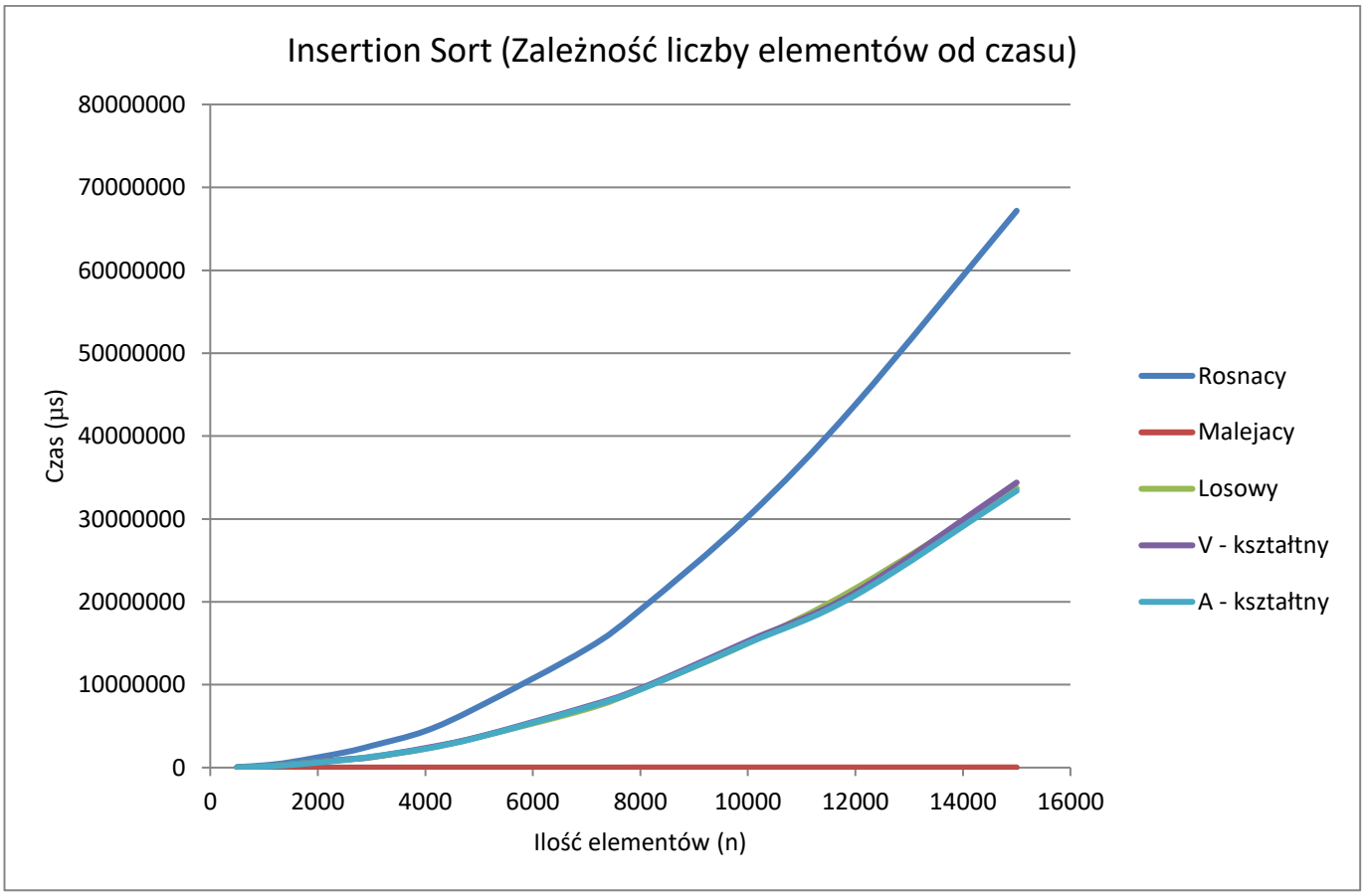
Heap Sort (Zależność liczby elementów od czasu)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	3015	3967	4011	5994	4018
1000	13059	12001	13024	13008	13091
1500	18034	19504	18034	18038	19259
2500	25801	29983	30063	30003	31101
3000	34503	36012	37036	36048	37024
4000	47018	48011	46973	48988	46116
5000	65269	63035	65954	60953	61007
7000	81915	93803	88838	90012	87017
8000	98042	102869	102963	103995	99964
10000	122937	140936	135993	131928	135113
12000	151036	164936	172978	163956	152893
15000	195953	227031	227887	211973	195866



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Heap Sort czasowo jest tak samo wydajny dla wszystkich pięciu rodzajów danych wejściowych.

c) Insertion Sort

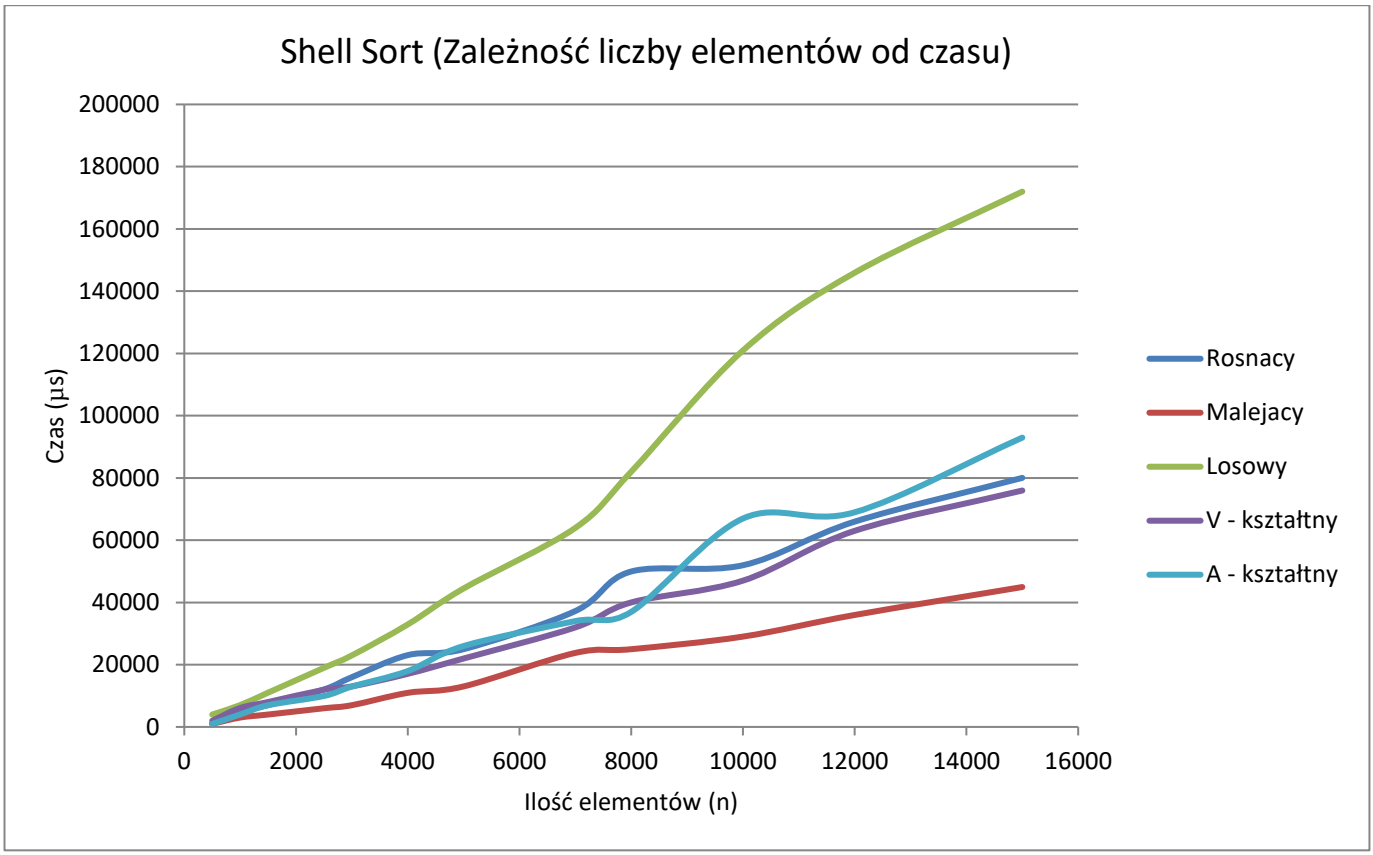
Insertion Sort (Zależność liczby elementów od czasu)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	64936	0	32980	32924	34986
1000	273234	1999	147242	140918	142187
1500	662591	6017	393801	361832	331768
2500	1844938	7029	968504	977479	912419
3000	2626750	7999	1252381	1275737	1315392
4000	4415528	7021	2275827	2357671	2272961
5000	7376614	6995	3707923	3721877	3667953
7000	14312924	7036	7048849	7360639	7246376
8000	19042504	12978	9450020	9548640	9426687
10000	30244194	13689	15014744	15264931	15010497
12000	43797788	13989	21614165	21158225	20785738
15000	67180005	18706	33707483	34389504	33397008



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Insertion Sort czasowo jest najbardziej wydajny dla danych w postaci malejącej, natomiast najmniej wydajny dla danych w postaci rosnącej.

d) Shell Sort

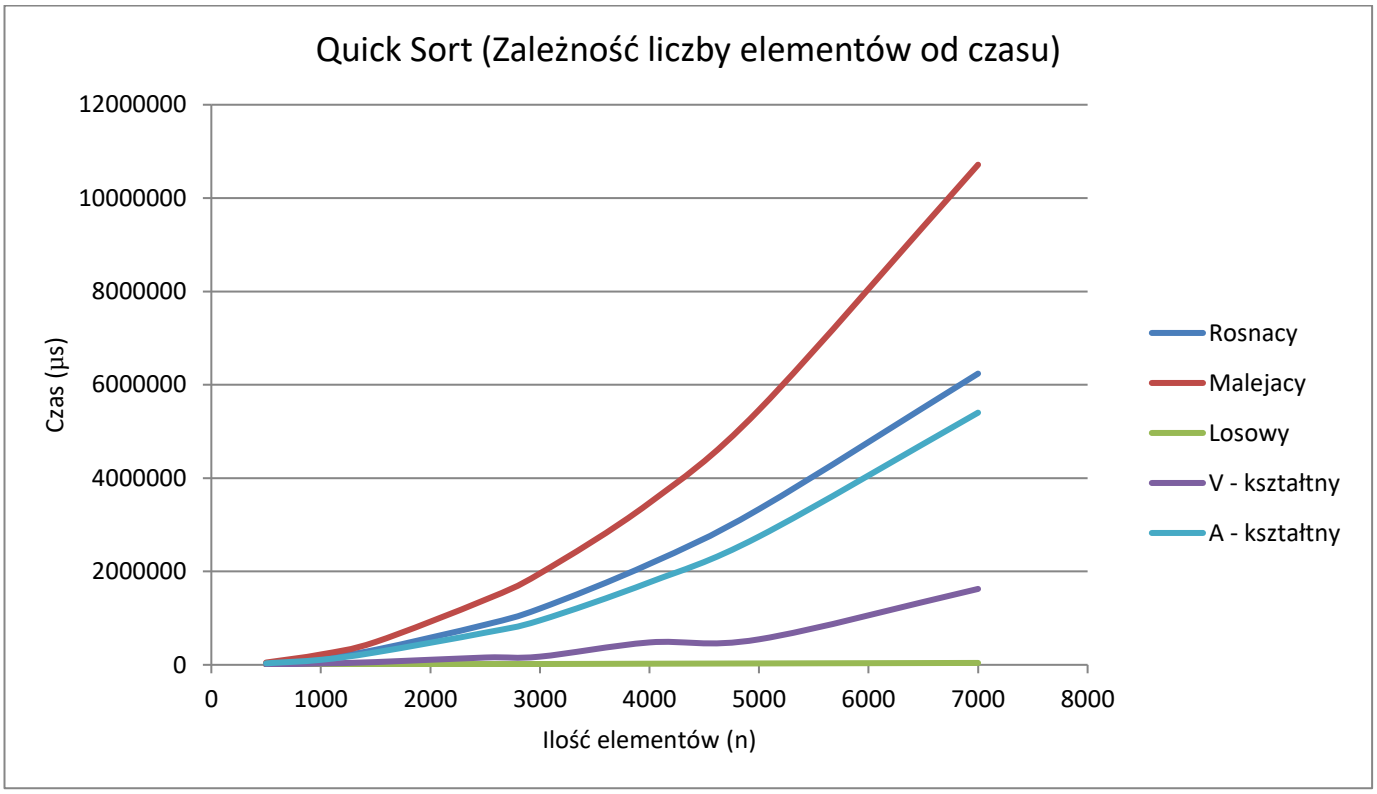
Shell Sort (Zależność liczby elementów od czasu)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	1032	1002	3979	1998	975
1000	4996	3020	7022	6076	3998
1500	7023	3988	11013	7996	6973
2500	12045	5999	18989	11949	9953
3000	15988	6991	22987	13028	12976
4000	23066	10955	32929	17055	17957
5000	24986	13017	44492	21987	25995
7000	37244	23787	63956	31982	34016
8000	49944	24975	82010	40007	36967
10000	51949	29025	120954	47001	66965
12000	65963	35978	145930	63004	68964
15000	80016	44952	171983	75959	92950



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Shell Sort czasowo jest najbardziej wydajny dla danych w postaci malejącej, natomiast najmniej wydajny dla danych w postaci losowej.

e) Quick Sort

Quick Sort (Zależność liczby elementów od czasu)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	32962	48952	3082	5983	31949
1000	145999	221844	7018	29042	107948
1500	322810	490720	13047	58927	267777
2500	860074	1394234	18007	156662	688728
3000	1206320	1958783	18092	175913	952561
4000	2162049	3470350	24042	480124	1768299
5000	3336326	5462017	30041	546775	2748618
7000	6240158	10715778	41048	1626095	5403427

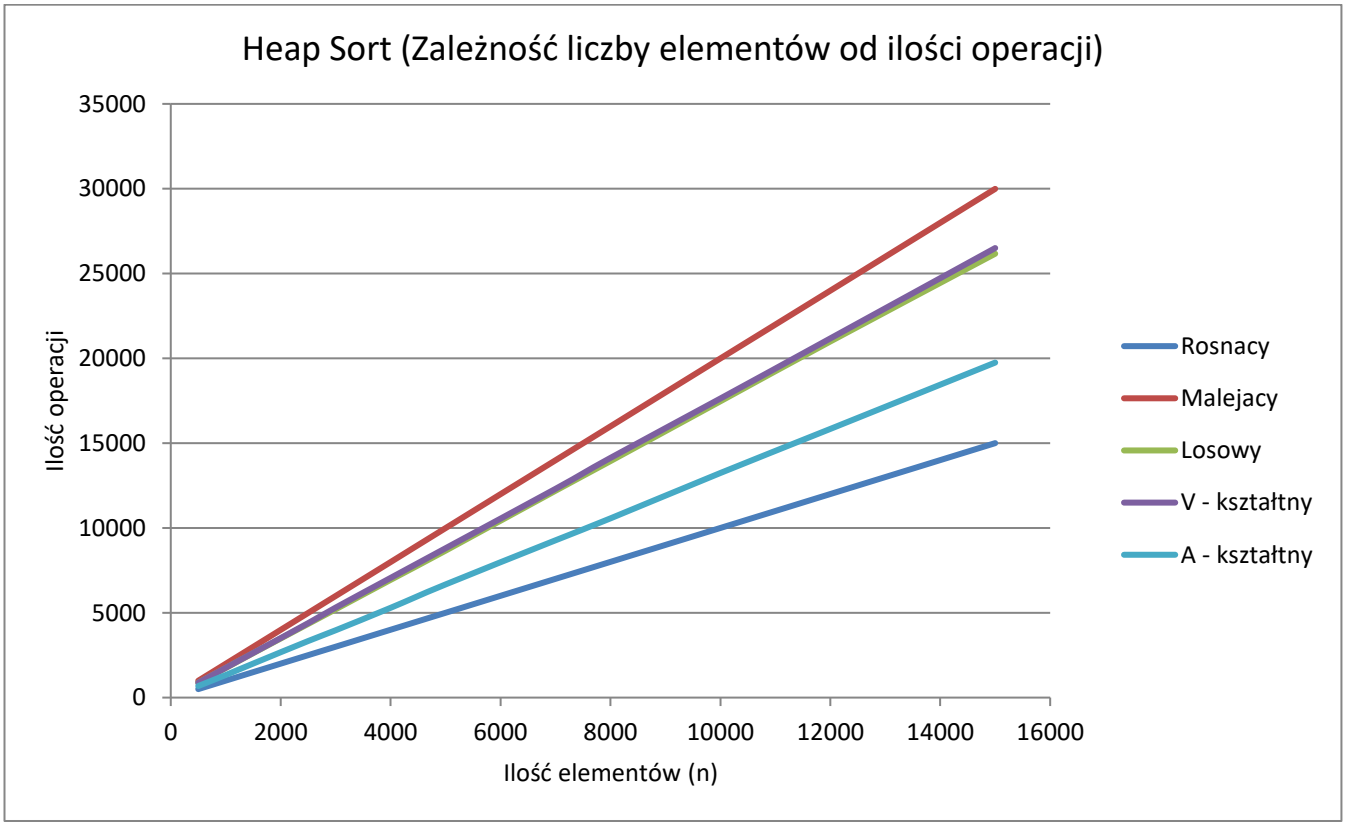


OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Quick Sort czasowo jest najbardziej wydajny dla danych w postaci losowej, natomiast najmniej wydajny dla danych w postaci malejącej.

2. Zależności liczby operacji od liczby elementów tablicy (n) *(Pokazując zależność algorytmów od danych wejściowych)*

   a) *Heap Sort*

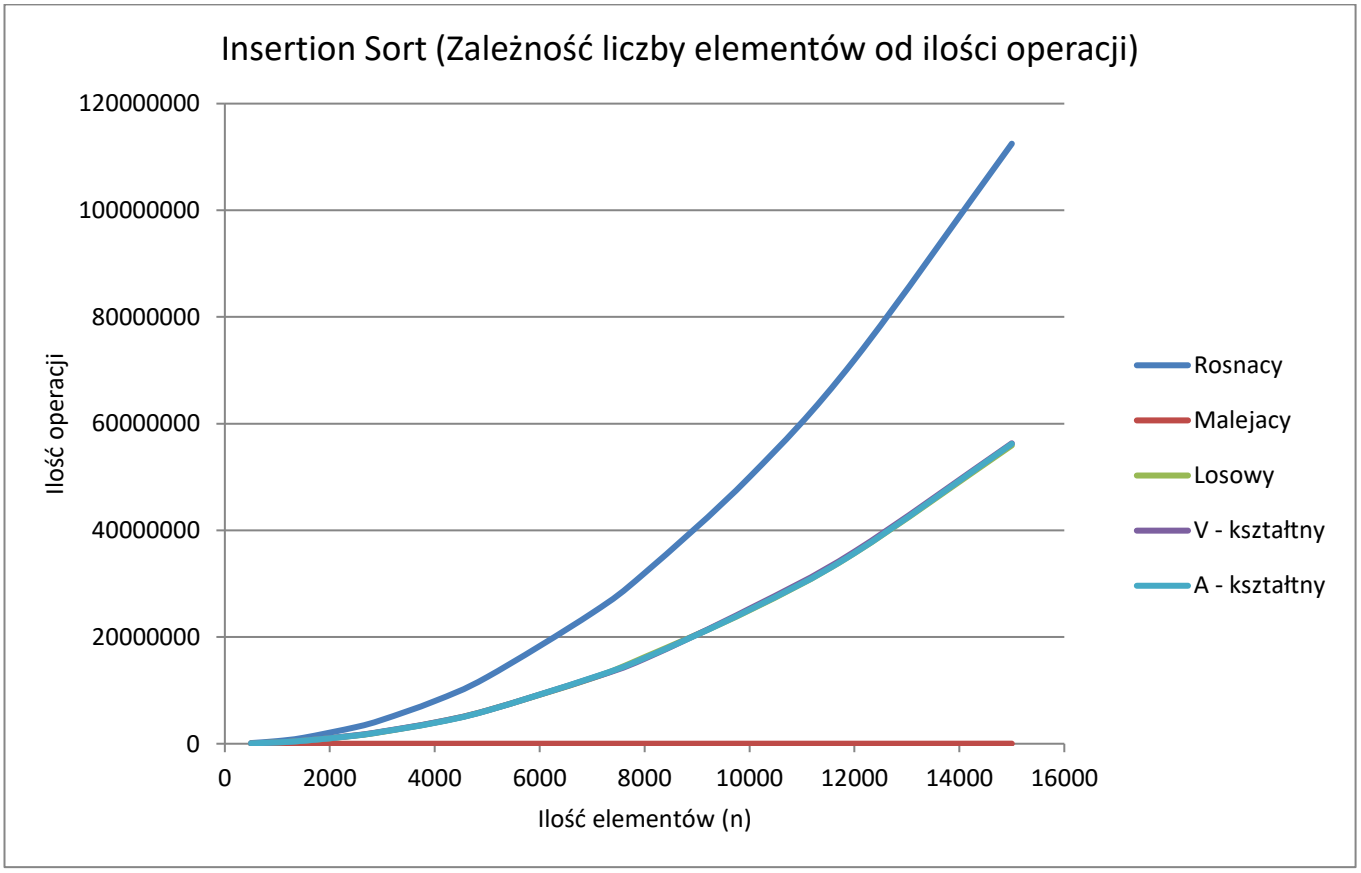
Heap Sort (Zależność liczby elementów od liczby operacji)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	500	993	876	875	681
1000	1000	1992	1746	1755	1331
1500	1500	2992	2622	2633	2006
2500	2500	4991	4344	4390	3334
3000	3000	5991	5207	5306	3970
4000	4000	7990	6941	7057	5292
5000	5000	9993	8663	8806	6672
7000	7000	13991	12195	12307	9273
8000	8000	15989	13942	14131	10565
10000	10000	19993	17468	17631	13240
12000	12000	23989	20994	21174	15838
15000	15000	29991	26168	26502	19750



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Heap Sort wykonuje największą ilość operacji porównań dla zbioru uporządkowanego w kolejności malejącej, natomiast najmniejszą ilość operacji wykonuje dla ciągu uporządkowanego rosnąco.

b) Insertion sort

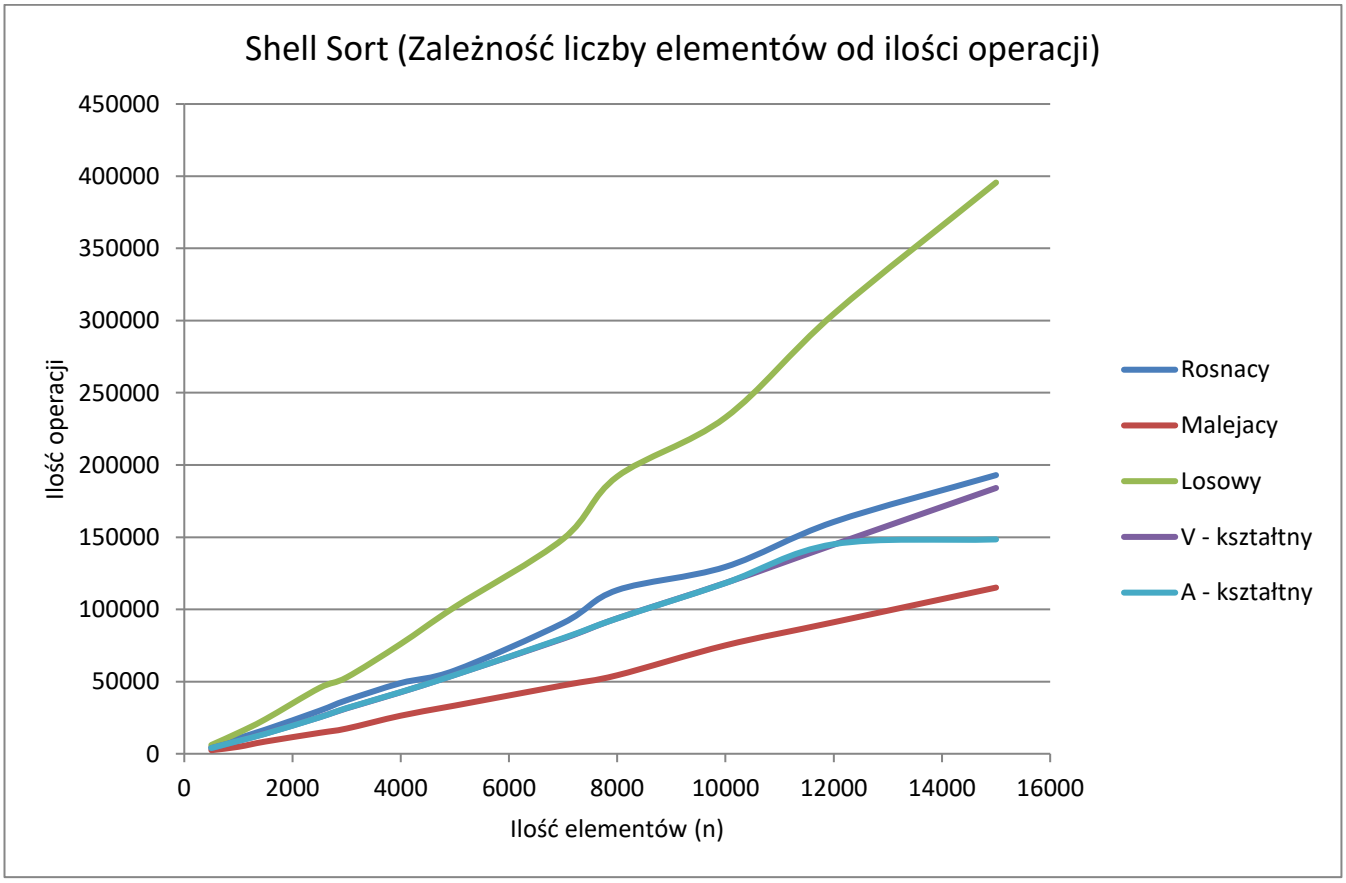
Insertion Sort (Ilość operacji)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	124750	499	62196	63497	68004
1000	499500	999	253383	247242	255686
1500	1124246	1499	560123	561803	572781
2500	3123743	2499	1539261	1585294	1537731
3000	4498483	2999	2227433	2290797	2250383
4000	7997945	3999	3958681	3976728	3917256
5000	12497436	4999	6243903	6235424	6208571
7000	24496260	6999	12290044	12325248	12377087
8000	31995670	7999	16236453	15932353	16062173
10000	49994378	9999	25025725	25312244	25103409
12000	71992912	11999	35719320	36009602	35739067
15000	112490509	14999	55938941	56328273	56214165



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Insertion Sort wykonuje największą ilość operacji porównań dla zbioru uporządkowanego w kolejności rosnącej, natomiast najmniejszą ilość operacji wykonuje dla ciągu uporządkowanego malejąco.

c) Shell sort

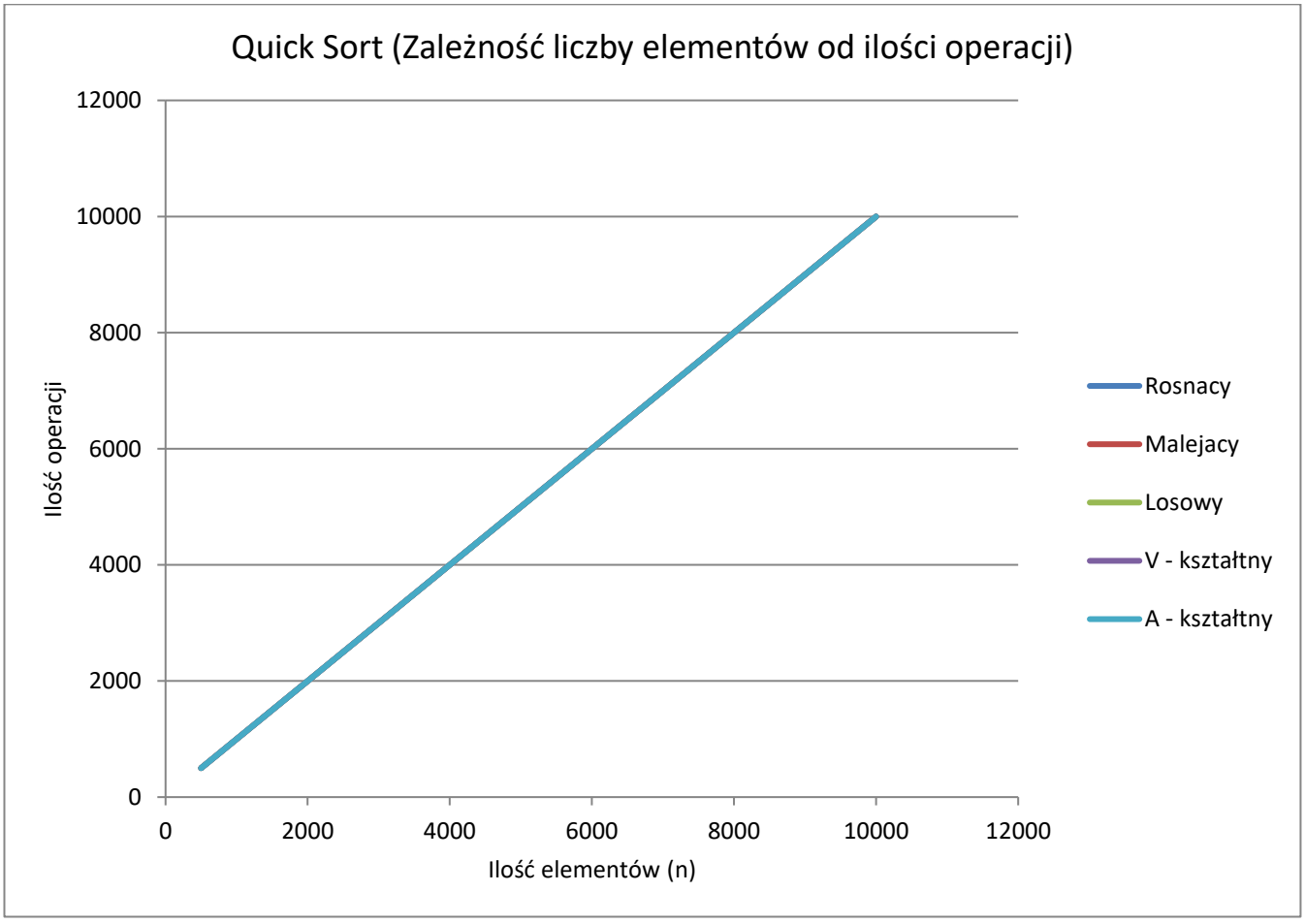
Shell Sort (Ilość operacji)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	4621	2457	6061	3899	3964
1000	10795	4821	14548	8957	8918
1500	16685	8457	23912	14018	13856
2500	29794	14457	45606	25291	25248
3000	37077	17457	52873	31451	31627
4000	48962	26364	76086	42626	42727
5000	57681	33364	101647	54659	54609
7000	90522	47364	148793	79693	80075
8000	113351	54364	191980	93777	93729
10000	129420	75084	232890	118271	118243
12000	160552	91084	304580	144769	145132
15000	192981	115084	395621	184066	148450



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Shell Sort wykonuje największą ilość operacji porównań dla zbioru uporządkowanego losowo, natomiast najmniejszą ilość operacji wykonuje dla ciągu uporządkowanego malejąco.

d) Quick sort

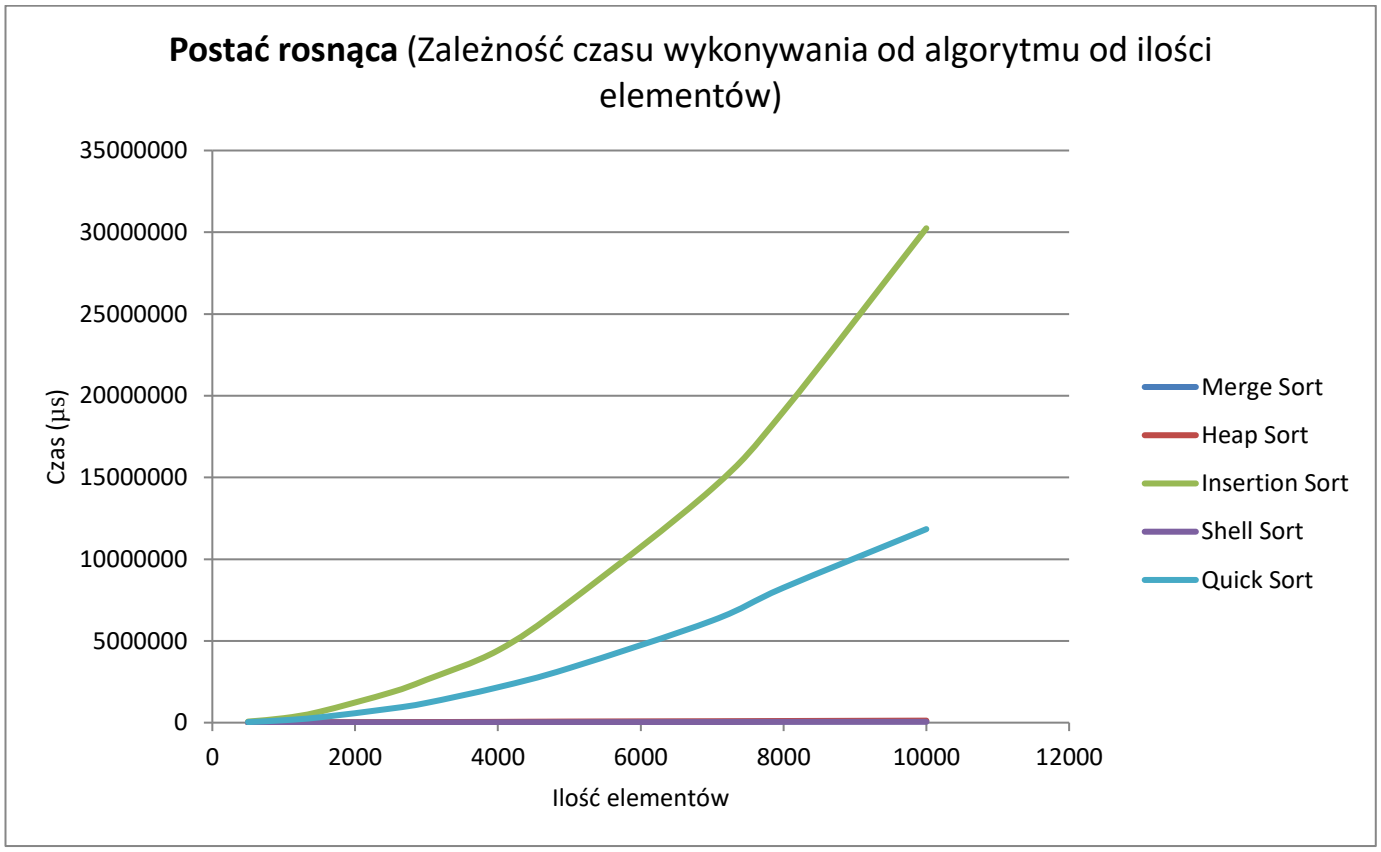
Quick Sort (Ilość operacji)					
	Rosnący	Malejący	Losowy	V - kształtny	A - kształtny
500	499	499	499	499	499
1000	999	999	999	999	999
1500	1499	1499	1499	1499	1499
2500	2499	2499	2499	2499	2499
3000	2999	2999	2999	2999	2999
4000	3999	3999	3999	3999	3999
5000	4999	4999	4999	4999	4999
7000	6999	6999	6999	6999	6999



OBSERWACJE: Ze zgromadzonych danych wynika, że algorytm sortowania Quick Sort wykonuje taką samą ilość obliczeń dla każdej postaci danych wejściowych.

3. Zależność czasu obliczeń od ilości elementów tablicy (Pokazują efektywność zaimplementowanych metod sortowania dla wybranej postaci danych).
   a) Ciąg rosnący

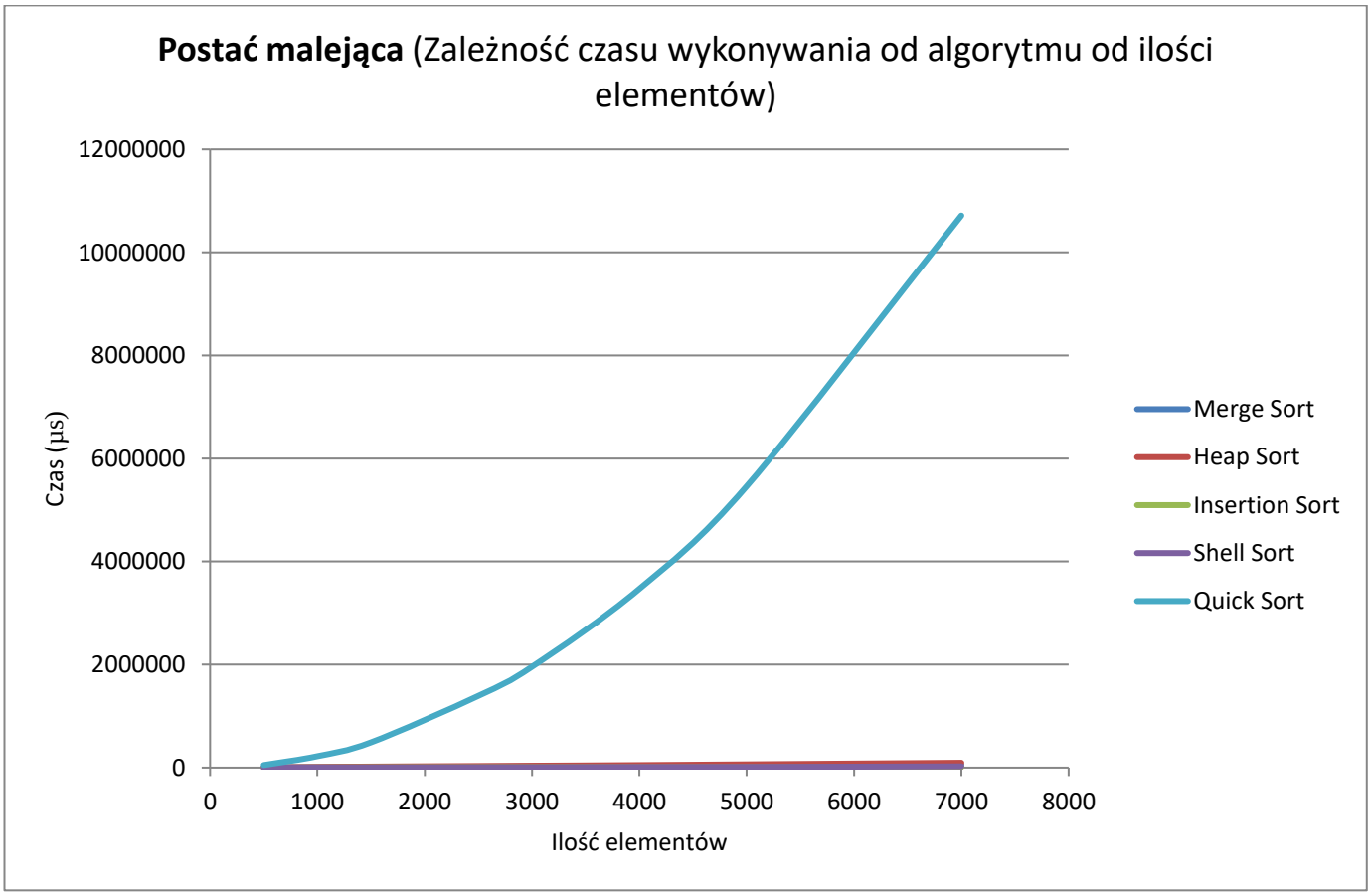
Ciąg rosnący					
	Merge Sort	Heap Sort	Insertion Sort	Shell Sort	Quick Sort
500	2999	3015	64936	1032	32962
1000	5981	13059	273234	4996	145999
1500	9951	18034	662591	7023	322810
2500	17973	25801	1844938	12045	860074
3000	22004	34503	2626750	15988	1206320
4000	30983	47018	4415528	23066	2162049
5000	41975	65269	7376614	24986	3336326
7000	56924	81915	14312924	37244	6240158
8000	67246	98042	19042504	49944	8254966
10000	86960	122937	30244194	51949	11837387



OBSERWACJE: Ze zgromadzonych danych wynika, że dane uporządkowane rosnąco najszybciej sortuje algorytm Shell Sort i podobnie Merge Sort, natomiast najdłużej Insertion Sort.

b) Ciąg malejący

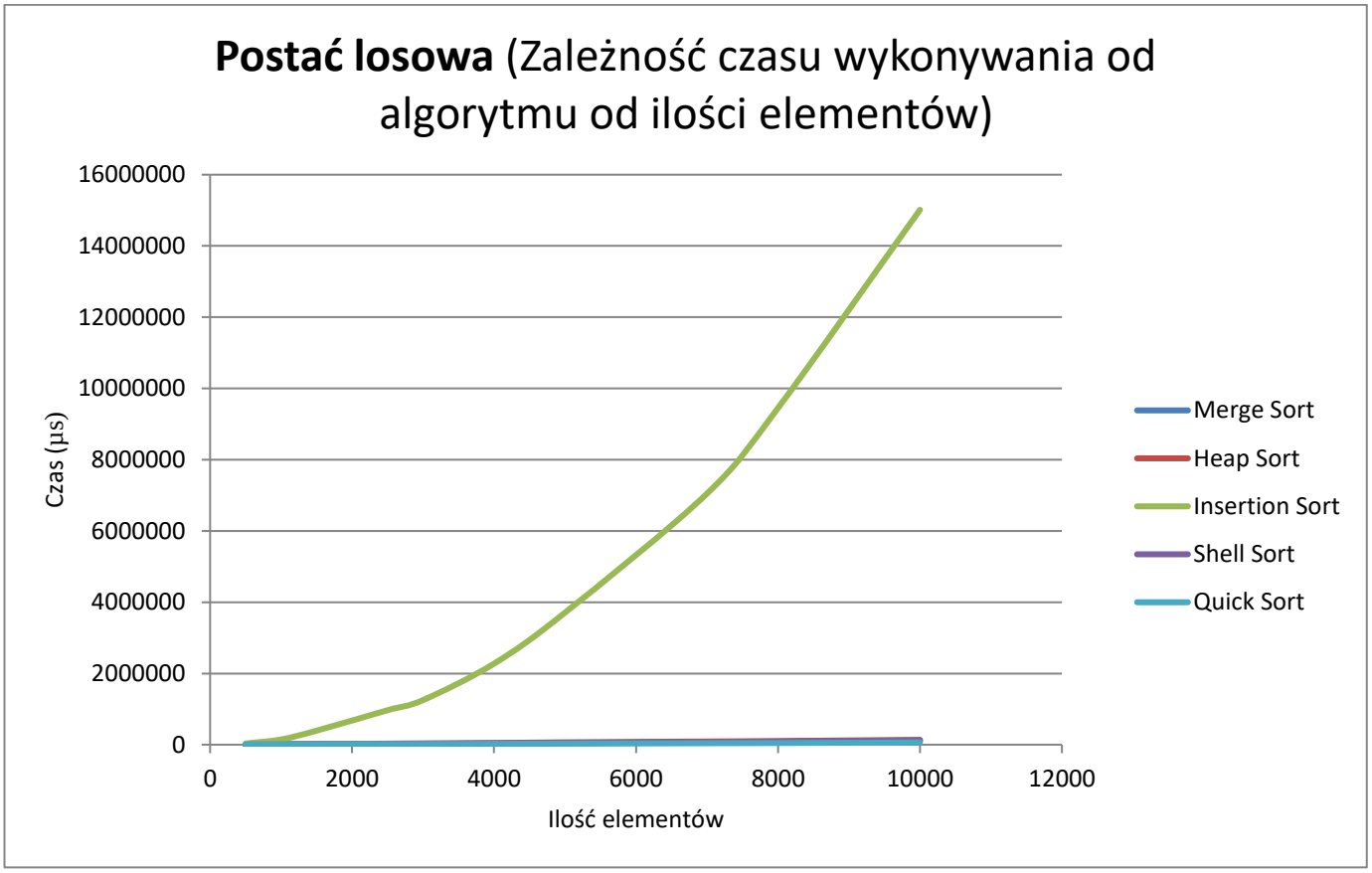
	Ciąg malejący				
	Merge Sort	Heap Sort	Insertion Sort	Shell Sort	Quick Sort
500	2999	3967	0	1002	48952
1000	7009	12001	1999	3020	221844
1500	11007	19504	6017	3988	490720
2500	17994	29983	7029	5999	1394234
3000	22986	36012	7999	6991	1958783
4000	31980	48011	7021	10955	3470350
5000	38937	63035	6995	13017	5462017
7000	55987	93803	7036	23787	10715778



OBSERWACJE: Ze zgromadzonych danych wynika, że dane uporządkowane malejąco najszybciej sortują algorytmy: Insertion Sort i następnie Shell Sort, Heap Sort i Merge Sort, natomiast najdłużej QuickSort.

c) Ciąg losowy

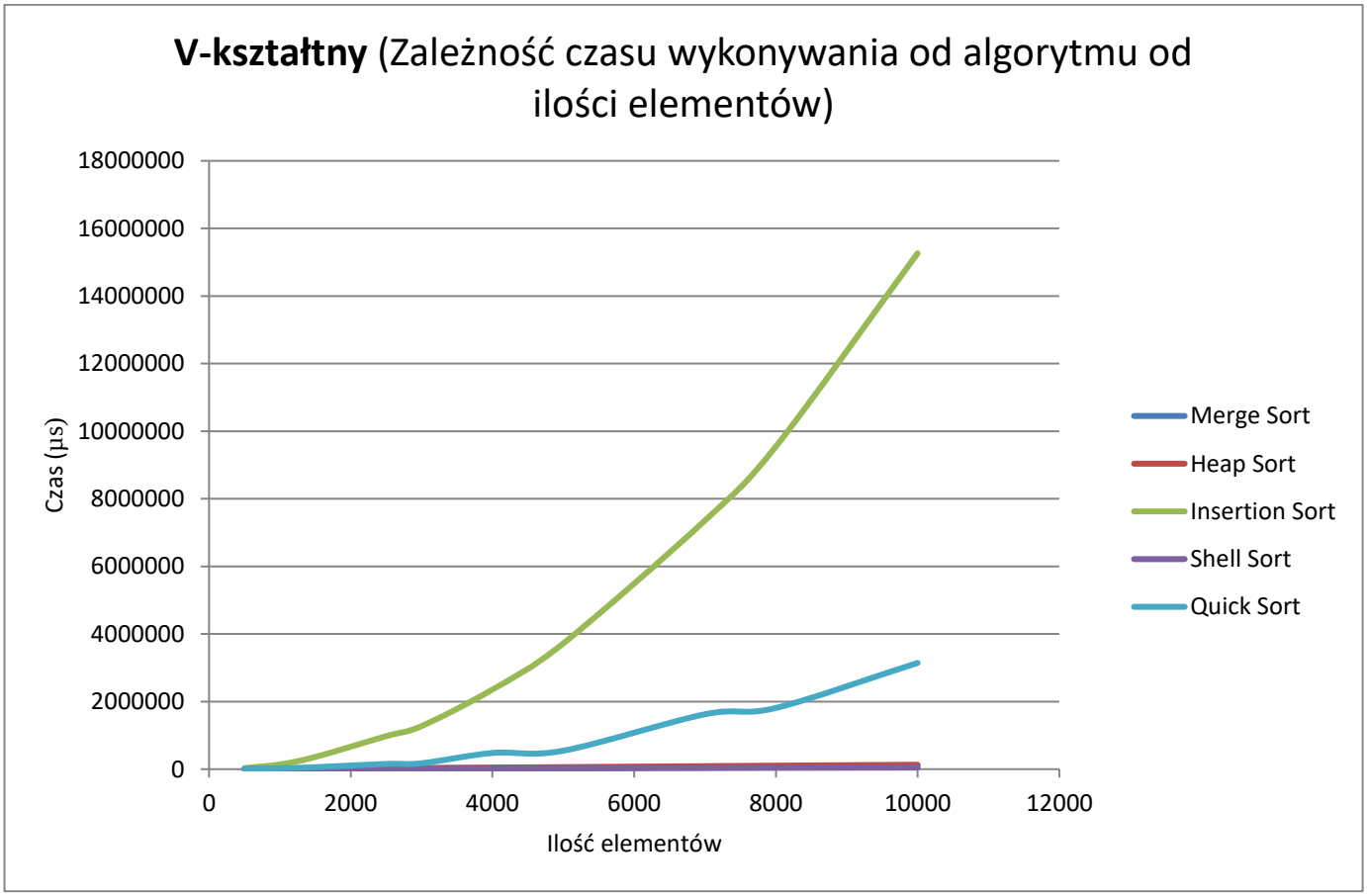
	Ciąg losowy				
	Merge Sort	Heap Sort	Insertion Sort	Shell Sort	Quick Sort
500	3011	4011	32980	3979	3082
1000	8015	13024	147242	7022	7018
1500	13022	18034	393801	11013	13047
2500	22941	30063	968504	18989	18007
3000	26941	37036	1252381	22987	18092
4000	38995	46973	2275827	32929	24042
5000	48930	65954	3707923	44492	30041
7000	71941	88838	7048849	63956	41048
8000	83970	102963	9450020	82010	46870
10000	107951	135993	15014744	120954	62479



OBSERWACJE: Ze zgromadzonych danych wynika, że dane uporządkowane malejąco najszybciej sortują algorytmy: Insertion Sort i następnie Shell Sort, Heap Sort i Merge Sort, natomiast najdłużej QuickSort.

d) Ciąg V-kształtny

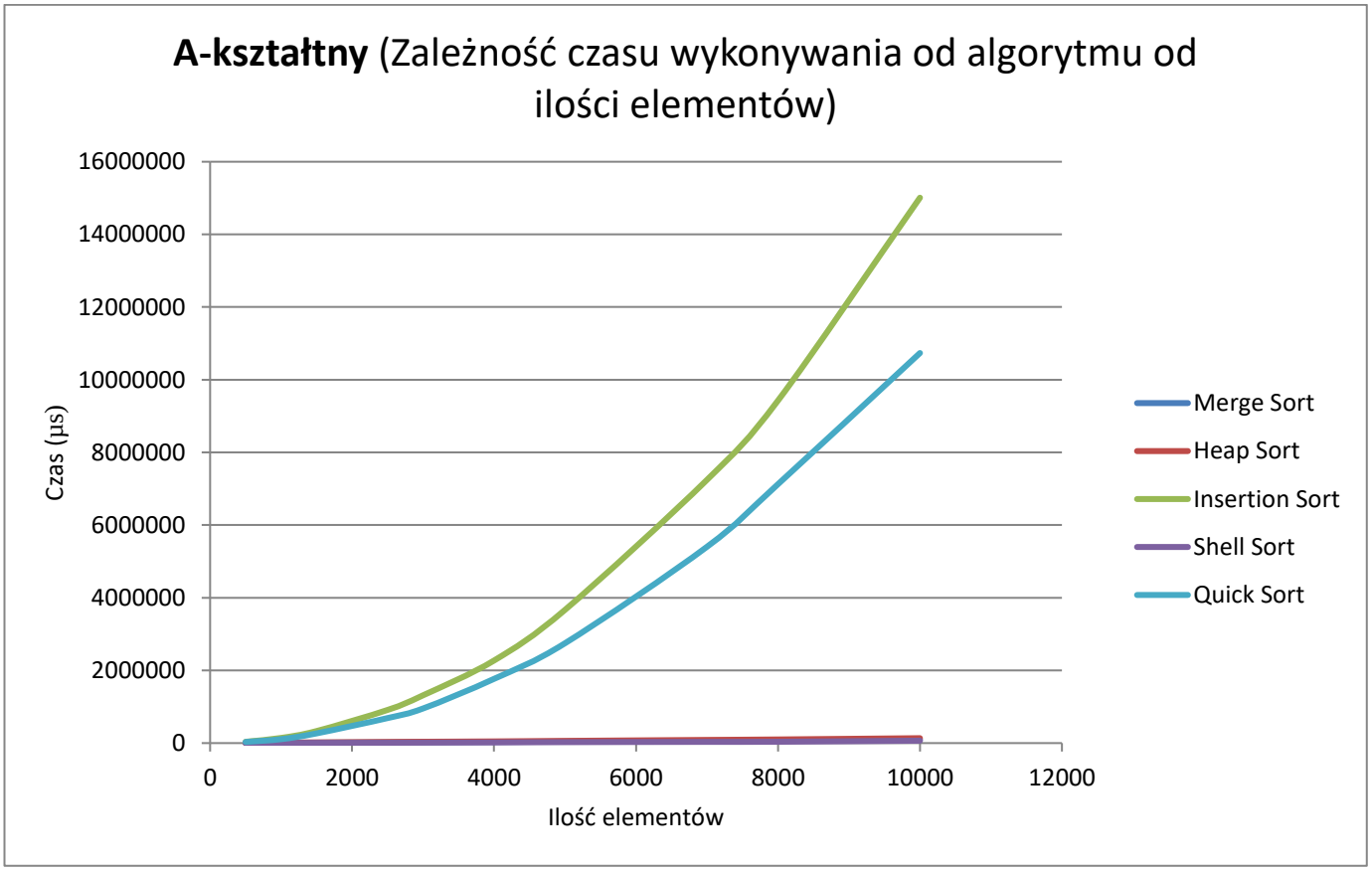
Ciąg V-kształtny					
	Merge Sort	Heap Sort	Insertion Sort	Shell Sort	Quick Sort
500	3018	5994	32924	1998	5983
1000	6997	13008	140918	6076	29042
1500	9968	18038	361832	7996	58927
2500	17990	30003	977479	11949	156662
3000	22987	36048	1275737	13028	175913
4000	32964	48988	2357671	17055	480124
5000	41451	60953	3721877	21987	546775
7000	58980	90012	7360639	31982	1626095
8000	68935	103995	9548640	40007	1812377
10000	89909	131928	15264931	47001	3144153



OBSERWACJE: Ze zgromadzonych danych wynika, że dane uporządkowane w porządku V-kształtnym najszybciej sortują algorytmy: Shell Sort i Merge Sort, natomiast najdłużej Insertion Sort.

e) Ciąg A-kształtny

Ciąg A-kształtny					
	Merge Sort	Heap Sort	Insertion Sort	Shell Sort	Quick Sort
500	3018	4018	34986	975	31949
1000	6016	13091	142187	3998	107948
1500	10934	19259	331768	6973	267777
2500	19963	31101	912419	9953	688728
3000	23963	37024	1315392	12976	952561
4000	31995	46116	2272961	17957	1768299
5000	40986	61007	3667953	25995	2748618
7000	58983	87017	7246376	34016	5403427
8000	68359	99964	9426687	36967	7125936
10000	89944	135113	15010497	66965	10735822



OBSERWACJE: Ze zgromadzonych danych wynika, że dane uporządkowane w porządku A-kształtnym najszybciej sortują algorytmy: Shell Sort i Merge Sort, natomiast najdłużej Insertion Sort.

4. Złożoność obliczeniowa dla poszczególnych algorytmów sortujących:

	Złożoność obliczeniowa		
Algorytm	**Optymistyczna**	**Średnia**	**Pesymistyczna**
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Heap Sort	O(n log n)	O(n log n)	O(n log n)
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$
Shell Sort	$O(n^{3/2})$	$O(n^{3/2})$	$O(n^{3/2})$
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$

Złożoność obliczeniowa danego algorytmu zależy od tego ile operacji ma on do wykonania. Im więcej operacji tym większa złożoność obliczeniowa.

a) Merge Sort – algorytm sortowania ma taką samą złożoność obliczeniową w przypadku optymistycznym, pesymistycznym i pesymistycznym, co potwierdzają wykonane przez nas testy. Niezależnie od postaci wejściowej liczba operacji dla algorytmu jest bardzo podobna.
b) Heap Sort - algorytm sortowania ma taką samą złożoność obliczeniową w przypadku optymistycznym, pesymistycznym i pesymistycznym, co potwierdzają wykonane przez nas testy. Niezależnie od postaci wejściowej liczba operacji dla algorytmu jest bardzo podobna.
c) Insertion Sort – w przypadku tego algorytmu złożoność obliczeniowa ściśle zależy od danych wejściowych. W przypadku danych w postaci malejącej (przypadek optymistyczny) algorytm porówna jedynie sąsiednie elementy ciągu i zakończy pracę co przekłada się na szybkość sortowania. W przypadku danych w postaci rosnącej algorytm będzie porównywał i zamieniał kolejno każdy element ciągu z wszystkimi poprzednimi co przekłada się na długi czas sortowania.
d) Shell Sort – algorytm sortowania ma podobną złożoność obliczeniową w każdym przypadku, co potwierdzają wykonane przez nas testy.
e) Quick Sort – algorytm sortowania dla postaci optymistycznej (czyli takiej gdzie równomiernie rozłożone jest prawdopodobieństwo wyboru elementu z tablicy) jest najbardziej wydajny, natomiast w przypadku pesymistycznym (ciąg rosnący lub malejący) złożoność obliczeniowa jest największa co potwierdzają wykonane przez nas testy.