Sprawozdanie

Autor: Sławomir Staniszewski

Nr albumu: 135893

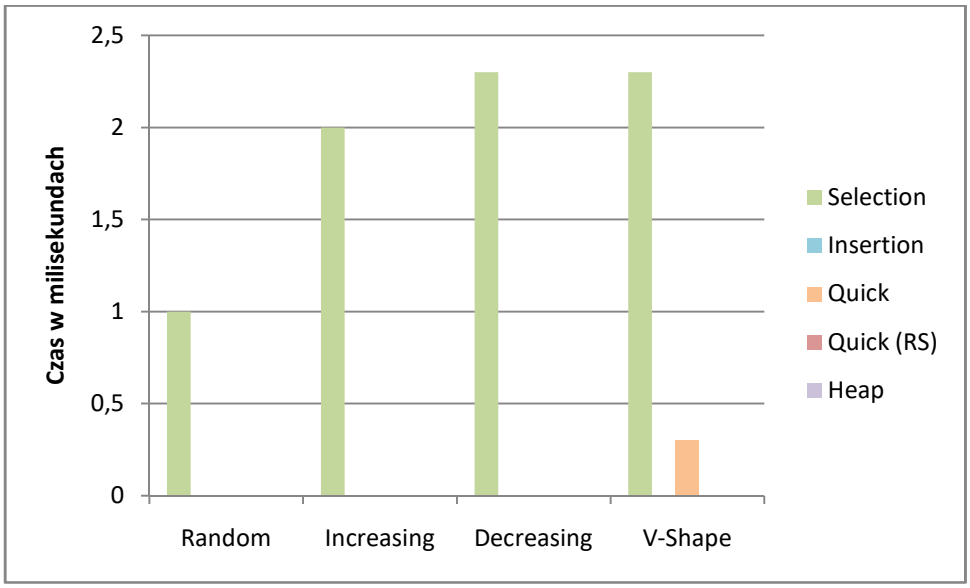
Grupa dziekańska: 3a

Algorytmy i Struktury Danych

Temat: Algorytmy sortowania

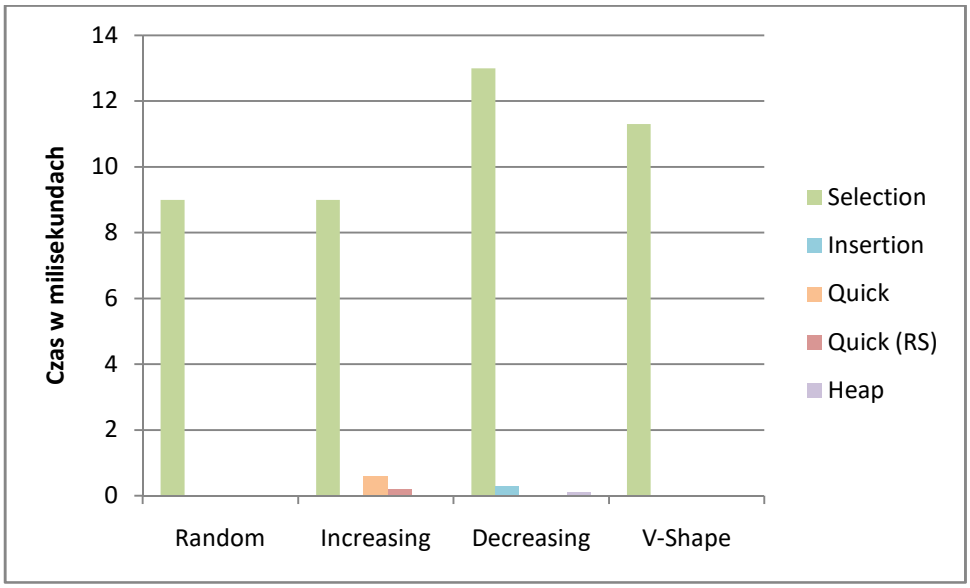
Pomiar czasu sortowania 100 elementów, wyrażony w milisekundach:

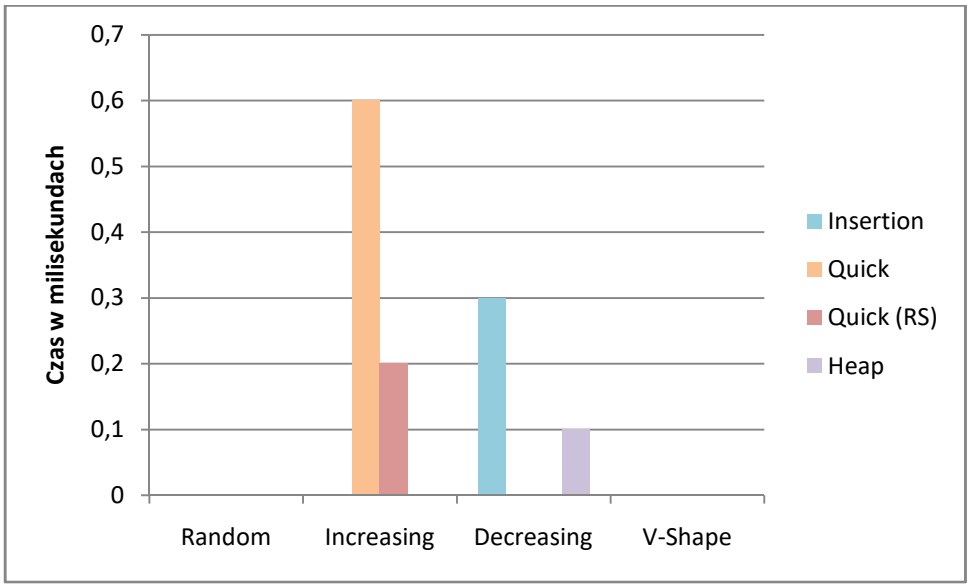
	100 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	1	2	2,3	2,3
Insertion	0	0	0	0
Quick	0	0	0	0,3
Quick (RS)	0	0	0	0
Heap	0	0	0	0



Pomiar czasu sortowania 200 elementów, wyrażony w milisekundach:

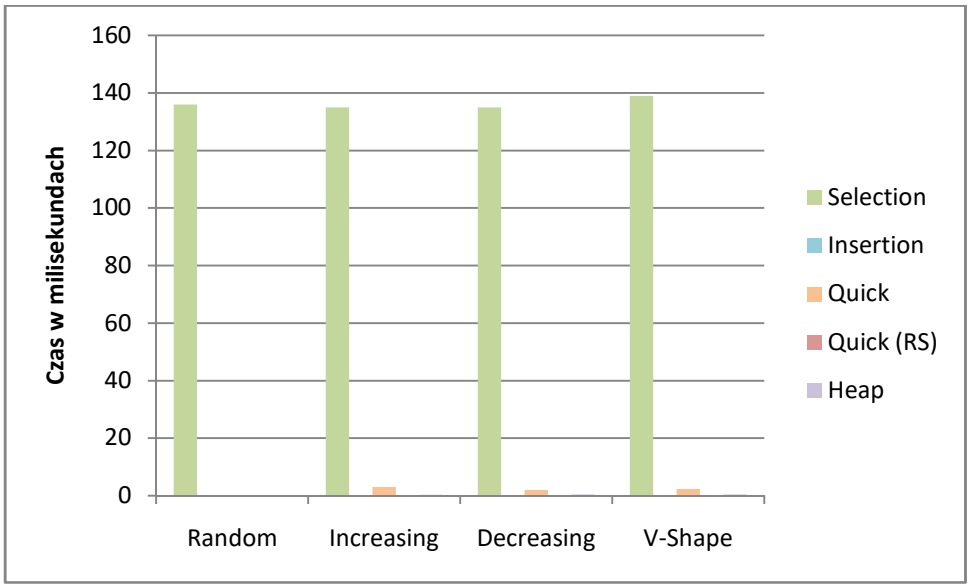
	200 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	9	9	13	11,3
Insertion	0	0	0,3	0
Quick	0	0,6	0	0
Quick (RS)	0	0,2	0	0
Heap	0	0	0,1	0

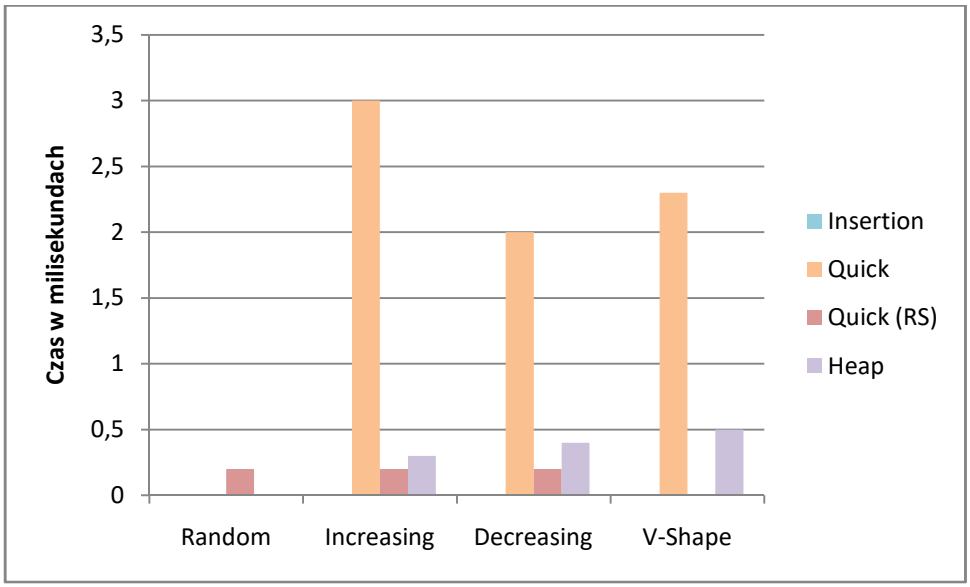




Pomiar czasu sortowania 500 elementów, wyrażony w milisekundach:

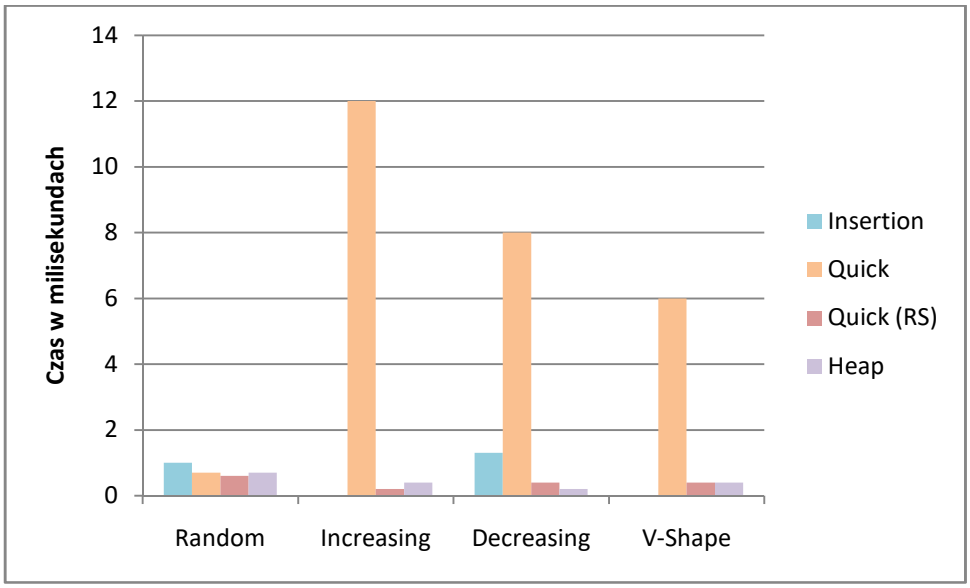
	500 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	136	135	135	139
Insertion	0	0	0	0
Quick	0	3	2	2,3
Quick (RS)	0,2	0,2	0,2	0
Heap	0	0,3	0,4	0,5





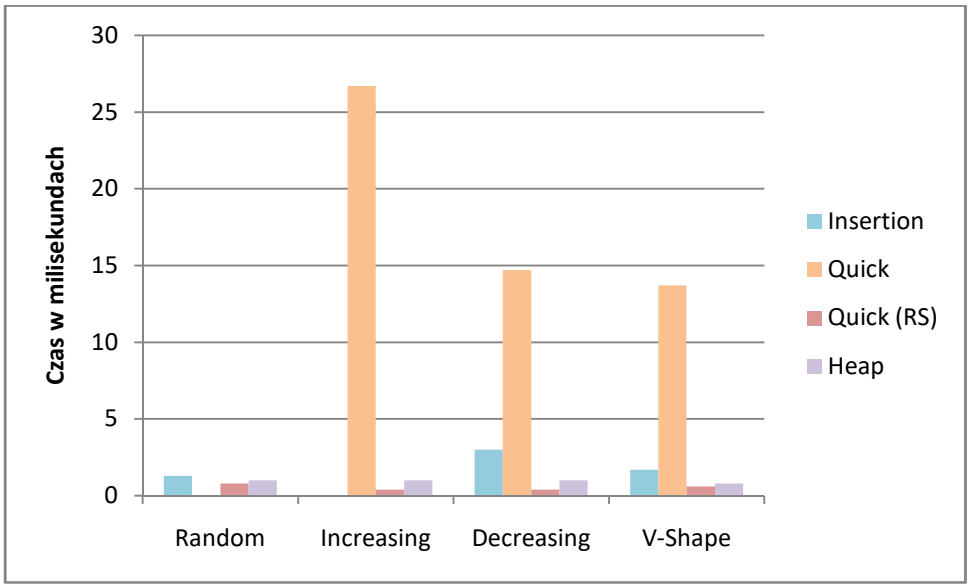
Pomiar czasu sortowania 1000 elementów, wyrażony w milisekundach:

	1000 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	1 079	1 070	1 068	1 068
Insertion	1	0	1,3	0
Quick	0,7	12	8	6
Quick (RS)	0,6	0,2	0,4	0,4
Heap	0,7	0,4	0,2	0,4



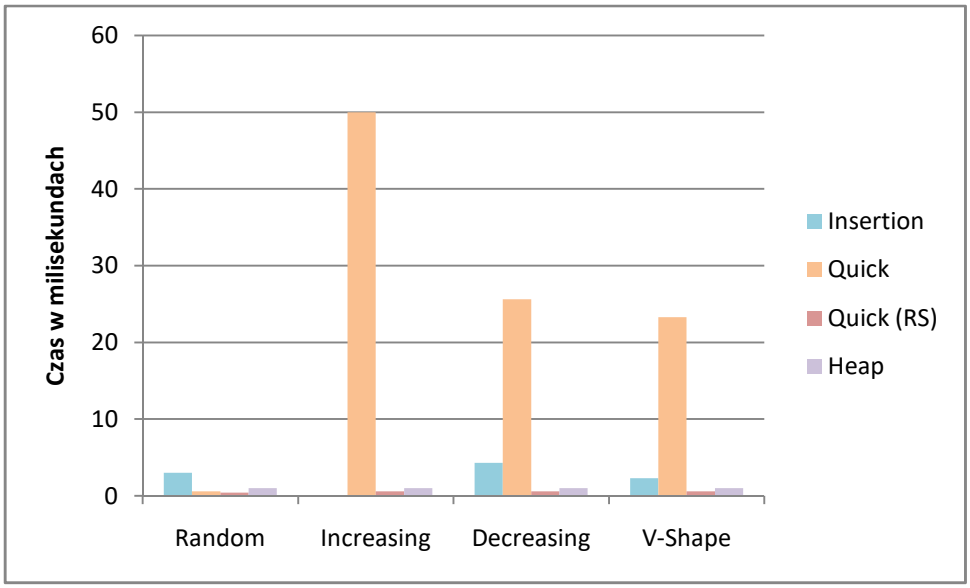
Pomiar czasu sortowania 1500 elementów, wyrażony w milisekundach:

	1500 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	3 618	3 612	3 603	3 594
Insertion	1,3	0	3	1,7
Quick	0	26,7	14,7	13,7
Quick (RS)	0,8	0,4	0,4	0,6
Heap	1	1	1	0,8



Pomiar czasu sortowania 2000 elementów, wyrażony w milisekundach:

	2000 objects			
	Random	Increasing	Decreasing	V-Shape
Selection	8 552	8 557	8 529	8 510
Insertion	3	0	4,3	2,3
Quick	0,6	50	25,6	23,3
Quick (RS)	0,4	0,6	0,6	0,6
Heap	1	1	1	1



Podsumowanie

Czas sortowania danych różnej wielkości przez zadane algorytmy, wyrażony w milisekundach:

		Quantity of objects					
		100	200	500	1000	1500	2000
Selection	Random	1	9	136	1 079	3 618	8 552
	Increasing	2	9	135	1 070	3 612	8 557
	Decreasing	2,3	13	135	1 068	3 603	8 529
	V-Shape	2,3	11,3	139	1 068	3 594	8 510
Insertion	Random	0	0	0	1	1,3	3
	Increasing	0	0	0	0	0	0
	Decreasing	0	0,3	0	1,3	3	4,3
	V-Shape	0	0	0	0	1,7	2,3
Quick Sort	Random	0	0	0	0,7	0	0,6
	Increasing	0	0,6	3	12	26,7	50
	Decreasing	0	0	2	8	14,7	25,6
	V-Shape	0,3	0	2,3	6	13,7	23,3
Quick Sort (Random Splitter)	Random	0	0	0,2	0,6	0,8	0,4
	Increasing	0	0,2	0,2	0,2	0,4	0,6
	Decreasing	0	0	0,2	0,4	0,4	0,6
	V-Shape	0	0	0	0,4	0,6	0,6
Heap Sort	Random	0	0	0	0,7	1	1
	Increasing	0	0	0,3	0,4	1	1
	Decreasing	0	0,1	0,4	0,2	1	1
	V-Shape	0	0	0,5	0,4	0,8	1

Pomiary wykonano na:
AMD Ryzen 7 2700X, 3,7 GHz, 8/16
16 GB RAM

Złożoność algorytmów:

	Przypadek	
	Najlepszy	*Najgorszy*
Selection	n^2	n^2
Insertion	n	n^2
Quick	$n\ log\ n$	n^2
Quick (RS)	$n\ log\ n$	n^2
Heap	$n\ log\ n$	$n\ log\ n$

Wnioski:

Selection sort – Czas sortowania rośnie bardzo szybko wraz ze wzrostem ilości danych.

Insertion sort – Działa najszybciej, jeśli dane są ułożone rosnąco.

Quick sort – Czas sortowania zależy od tego gdzie nastąpi podział tablicy. Trafianie cały czas w środek pozwala na uzyskanie złożoności logarytmicznej. Jeżeli dzielimy na skrajnych elementach to możemy osiągnąć najgorszą, kwadratową złożoność.

Quick sort (RS) – Dodanie losowej wartości jako elementu rozdzielającego pozwala zmniejszyć szanse na trafianie cały czas w skrajne elementy. Przyspiesza to znacząco szybkość działania algorytmu, co uwidacznia się w powyższych pomiarach.

Heap sort – Jest wolniejszy niż najbardziej optymistyczny przypadek Quick sort. Jego zaletą jest najprostsza złożoność obliczeniowa w najgorszym przypadku ułożenia danych.