Algorytmy sortowania

Sprawozdanie

Autorzy:
Marek Prywer 146 425
Radosław Stefański 146 544

1. Opis

Celem niniejszego sprawozdania jest przeanalizowanie algorytmów sortowania, a konkretniej ich zachowania pod kątem wprowadzanych danych. Będziemy badać następujące algorytmy sortowania:

- selection sort
- insertion sort
- quick sort
- heap sort.

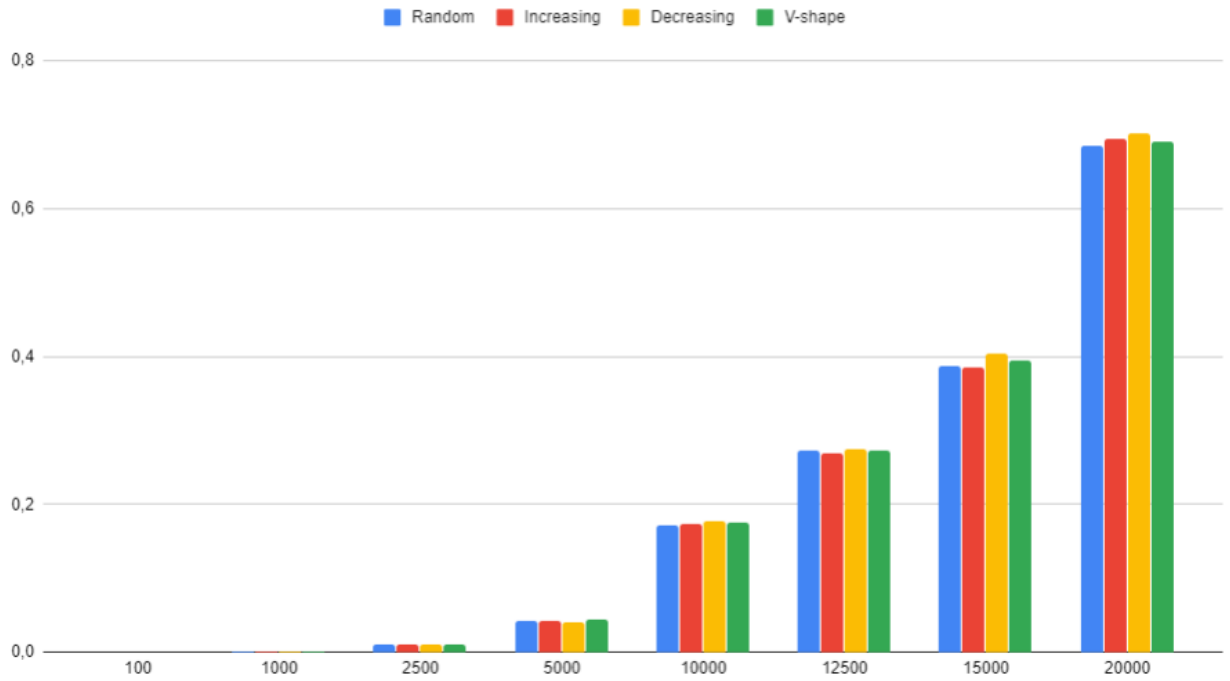
Do powyższych algorymtów będziemy stosować następujące dane wsadowe:

- liczby losowe
- liczby rosnące
- liczby malejące
- liczby których wartości układają się w kształ litery V (v-shape)

Obiektem naszych rozważań będzie oczywiście czas wykonywania skryptu w zależności od rozmiaru i układu danych wejściowych. Wszystkie algorytmy będziemy badać dla tyhch samych rozmiarów danych.

2. Algorytm Selection Sort

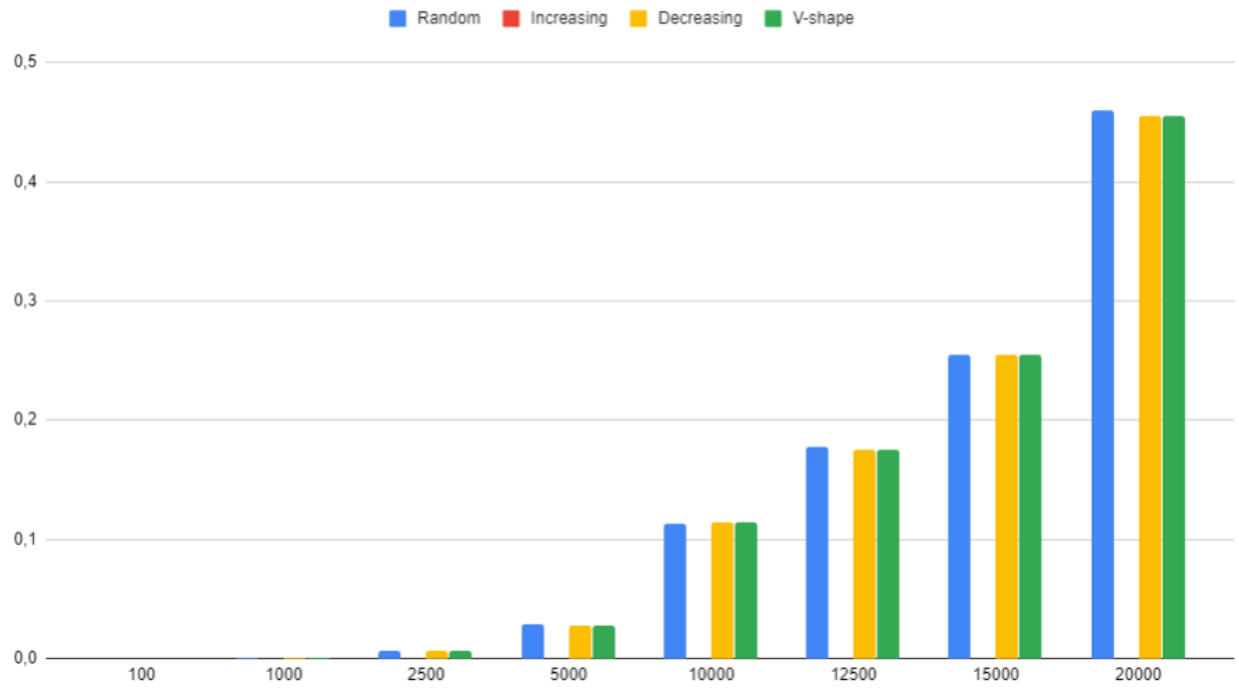
Rozmiar/typ	Random	Increasing	Decreasing	V-shape
100	0.000034	0.000030	0.000033	0.000026
1000	0.001849	0.001891	0.001890	0.001967
2500	0.011013	0.010891	0.011080	0.011008
5000	0.042394	0.043355	0.041554	0.043834
10000	0.171491	0.173014	0.176727	0.174659
12500	0.272038	0.269913	0.275576	0.273610
15000	0.387240	0.385475	0.403171	0.394599
20000	0.684991	0.693677	0.702260	0.691129



Okazuje się, że sortowanie przez wybieranie zachowuje się bardzo podobnie dla tych samych rozmarów danych wejściowych niezależnie od sposobu uszeregowania danych. Czas sortowania rośnie tak samo dla wszystkich typów danych tak samo.

3. Algorytm Insertion Sort

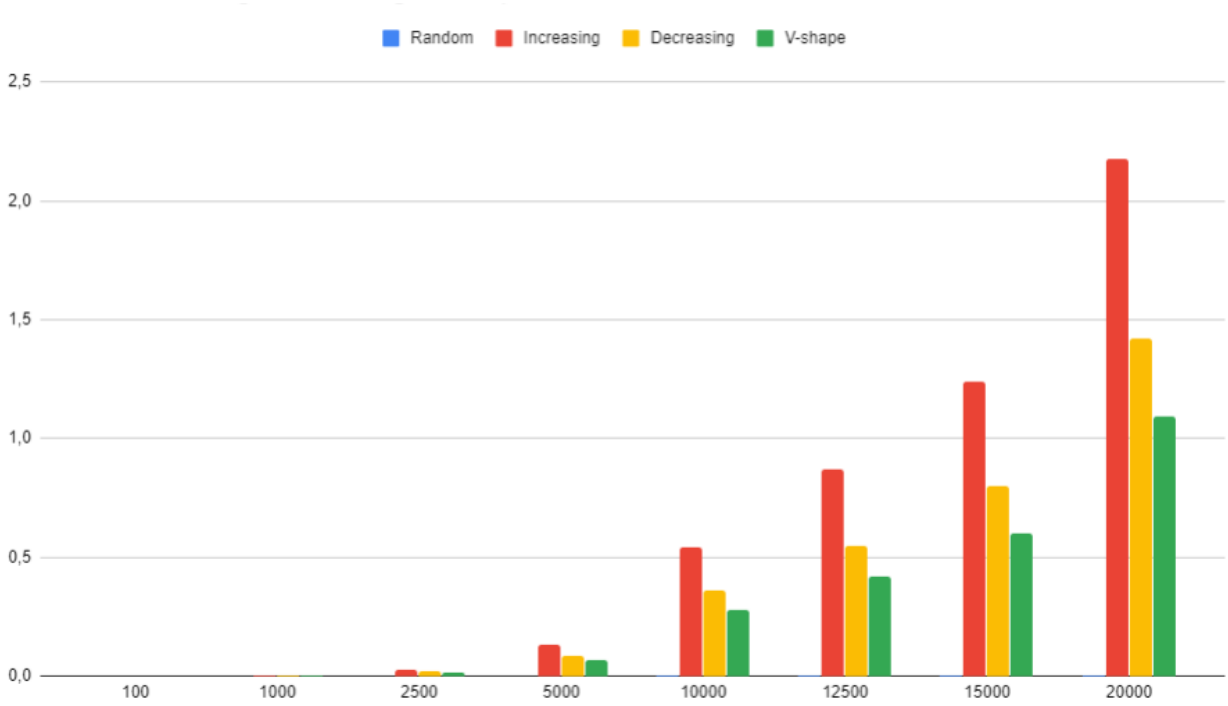
Rozmiar/typ	Random	Increasing	Decreasing	V-shape
100	0.000013	0.000002	0.000016	0.000016
1000	0.001161	0.000007	0.001149	0.001149
2500	0.007107	0.000018	0.007177	0.007177
5000	0.028977	0.000032	0.028316	0.028316
10000	0.113395	0.000071	0.114886	0.114886
12500	0.177135	0.000077	0.175602	0.175602
15000	0.254552	0.000102	0.254754	0.254754
20000	0.459271	0.000133	0.454659	0.454659



W przypadku sortowania przez wstawianie dane typu losowe, malejące oraz v-shape są bardzo podobnie. Wyjątkiem są natomiast dane rosnące. W tym przypadku czas wykonywania jest tak mały, że w zasadzie jest niewidoczny na powyższym wykresie.

4. Algorytm QuickSort

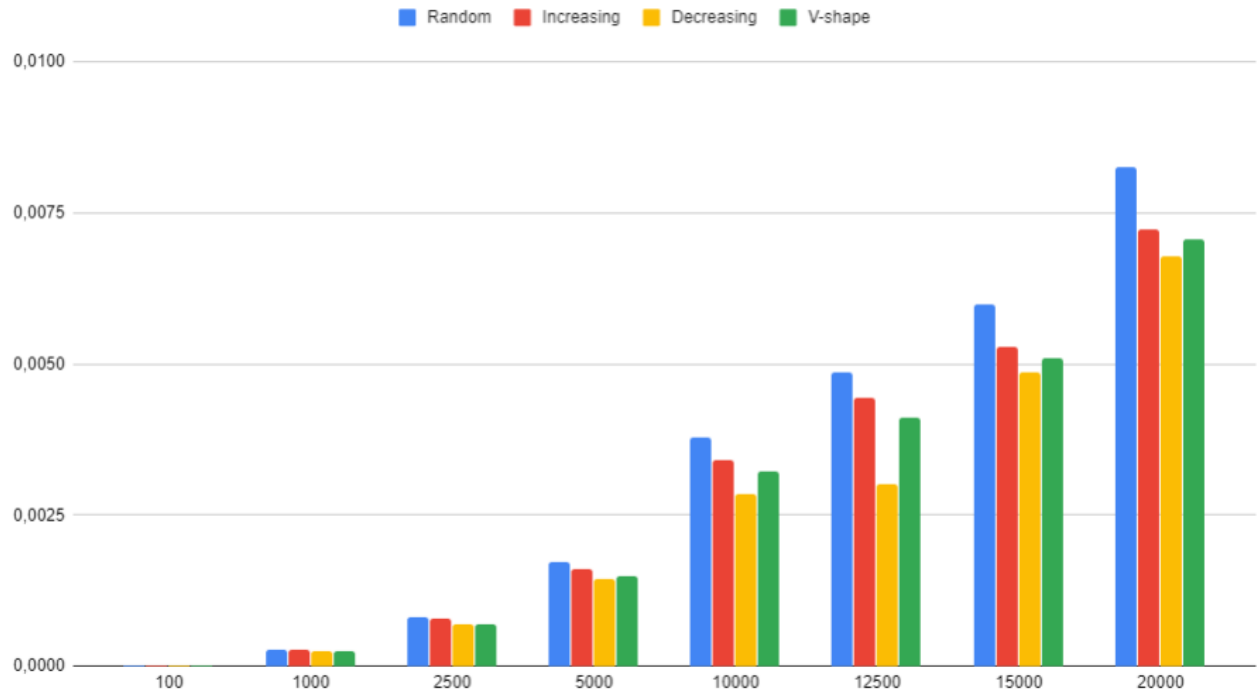
Rozmiar/typ	Random	Increasing	Decreasing	V-shape
100	0.000013	0.000055	0.000044	0.000041
1000	0.000143	0.005006	0.004000	0.002774
2500	0.000453	0.029418	0.023710	0.016819
5000	0.000943	0.131983	0.087474	0.065571
10000	0.002005	0.543323	0.361059	0.277691
12500	0.002541	0.869406	0.547748	0.420259
15000	0.003102	1.236486	0.801844	0.603040
20000	0.004675	2.174441	1.420601	1.093299



Algorytm Quick Sort poradził sobie najlepiej z sortowaniem danych w postaci losowej. Podobnie jak przy selection sort oraz danych rosnących – nie jest to widoczne na wykresie.

5. Algorytm Heap Sort

Rozmiar/typ	Random	Increasing	Decreasing	V-shape
100	0.000020	0.000020	0.000017	0.000018
1000	0.000273	0.000263	0.000251	0.000243
2500	0.000801	0.000778	0.000694	0.000688
5000	0.001719	0.001609	0.001451	0.001483
10000	0.003795	0.003402	0.002854	0.003232
12500	0.004867	0.004441	0.003022	0.004124
15000	0.005980	0.005275	0.004871	0.005093
20000	0.008267	0.007231	0.006780	0.007070



Algorytm heapsort okazał się najszybszym spośród wszystkich testowanych. Najlepiej poradził sobie z danymi malejącymi, a najgorzej z danymi losowymi, bo te nie są w żadnej części uporządkowane .