Laboratorium Algorytmy i Struktury Danych

Politechnika Poznańska

Temat: Algorytmy Sortowania

Informatyka zaoczna grupa 3b

Autorzy: Zuzanna Śliwińska, Adam Włodarczyk

1. **Tabele z czasami działania poszczególnych algorytmów w zależności od typu danych wejściowych i wielkości instancji**

	SelectionSort				InsertionSort			
n	Random	Increasing	Decreasing	V-Shape	Random	Increasing	Decreasing	V-Shape
100	0,000008	0,000005	0,000006	0,000007	0,000003	0,000001	0,000004	0,000005
200	0,000025	0,000016	0,000018	0,000021	0,000009	0,000001	0,000025	0,000014
300	0,000054	0,000035	0,00004	0,00006	0,000017	0,000001	0,000028	0,000029
400	0,000078	0,000061	0,000072	0,000074	0,000046	0,000001	0,000049	0,000047
500	0,000116	0,000097	0,00011	0,000121	0,000041	0,000001	0,000075	0,000068
600	0,000168	0,000134	0,000156	0,000171	0,000059	0,000001	0,000108	0,000098
700	0,000224	0,000182	0,000208	0,000247	0,00008	0,000001	0,000179	0,000152
800	0,000345	0,000236	0,000279	0,000276	0,000112	0,000002	0,00019	0,000145
900	0,000557	0,000301	0,000341	0,000355	0,00012	0,000002	0,00024	0,000175
1000	0,000526	0,00037	0,000413	0,000443	0,000155	0,000002	0,000295	0,000192

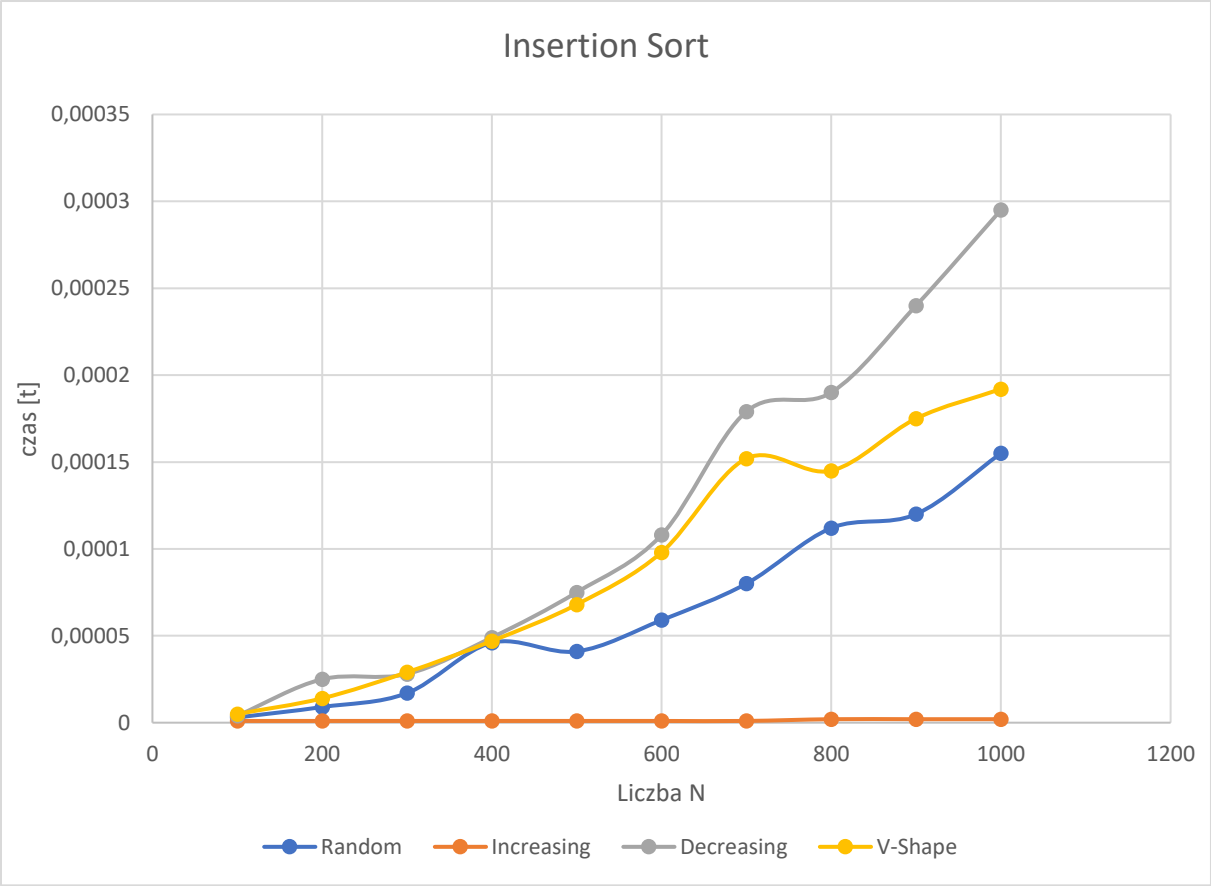
	QuickSort				HeapSort			
n	Random	Increasing	Decreasing	V-Shape	Random	Increasing	Decreasing	V-Shape
100	0,000005	0,000011	0,000009	0,000006	0,000006	0,000005	0,000005	0,000004
200	0,000011	0,000039	0,000037	0,000023	0,00001	0,000012	0,00001	0,000009
300	0,000016	0,000081	0,000069	0,000051	0,00002	0,000018	0,000018	0,000013
400	0,000022	0,000147	0,000114	0,000089	0,000027	0,000024	0,000023	0,000018
500	0,000028	0,000225	0,000183	0,000144	0,000034	0,000031	0,000025	0,000021
600	0,000034	0,000301	0,000261	0,000198	0,000042	0,00004	0,000031	0,00003
700	0,000041	0,000418	0,000358	0,000268	0,000048	0,000053	0,000037	0,000042
800	0,000047	0,000551	0,000453	0,00035	0,000057	0,000056	0,000044	0,000049
900	0,000053	0,000732	0,000597	0,000485	0,000066	0,000063	0,000052	0,000054
1000	0,000062	0,000891	0,000701	0,000546	0,000075	0,000067	0,000062	0,000057

2. **Wykresy korelacji algorytmów sortowania a rodzaju danych wejściowych i jej ilości.**

Złożoność obliczeniowa:

Przypadek		
Najlepszy	Średni	Najgorszy
$O(n^2)$	$O(n^2)$	$O(n^2)$

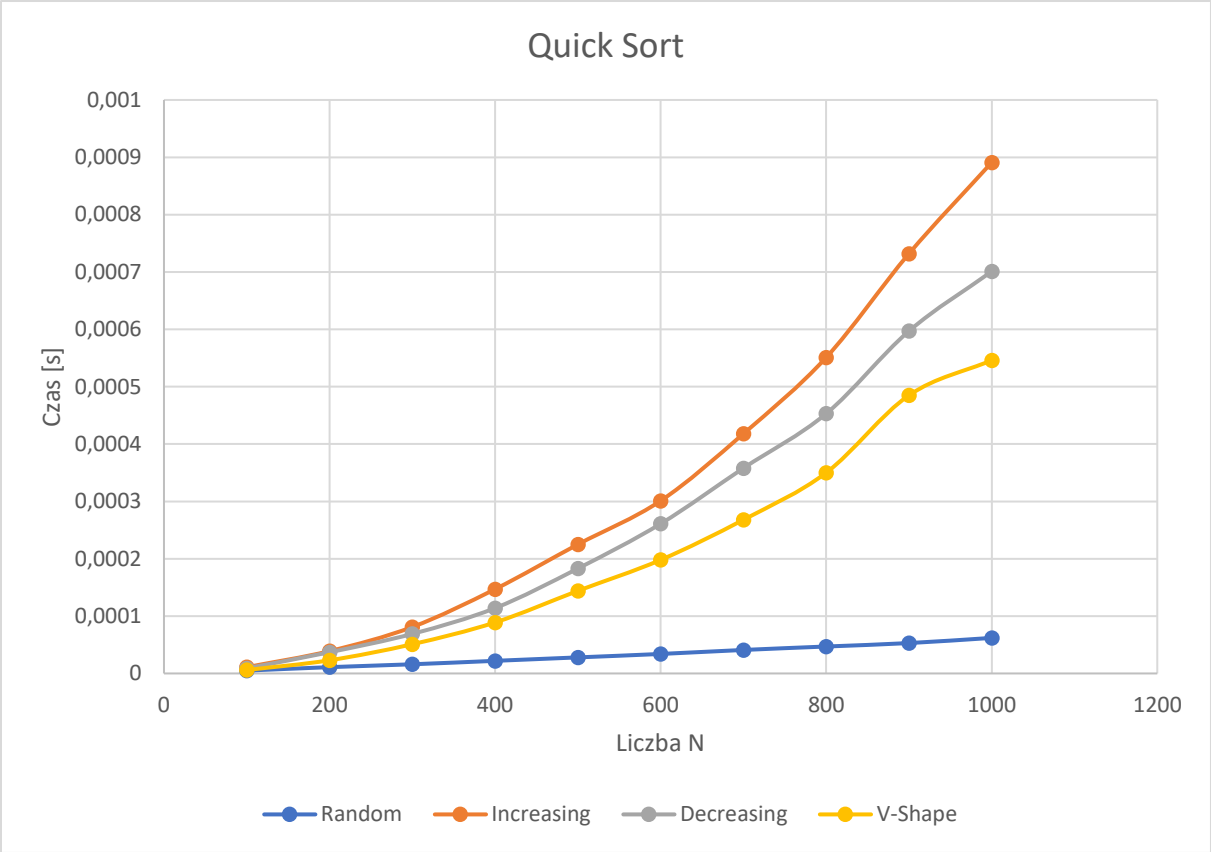
Algorytm ten ma stałą złożoność obliczeniową dla wszystkich przypadków, jednak czas jego realizacji nie jest zadawalający. Jedynym plusem tego algorytmu jest łatwość implementacji przez intuicyjność. Odbiegające od reguły zachowanie czasu realizacji algorytmu dla losowych danych wejściowych prawdopodobnie wynika z nagłego obciążenia maszyny obliczeniowej innym zadaniem. Aby uzyskać dokładniejsze wyniki powinno się wykonać obliczenia dla większej ilości danych lub powtórzyć kilkukrotnie operację i ją uśrednić.



Złożoność obliczeniowa:

Przypadek		
Najlepszy	Średni	Najgorszy
$O(n)$	$O(nlogn)$….$O(n^2)$	$O(n^2)$

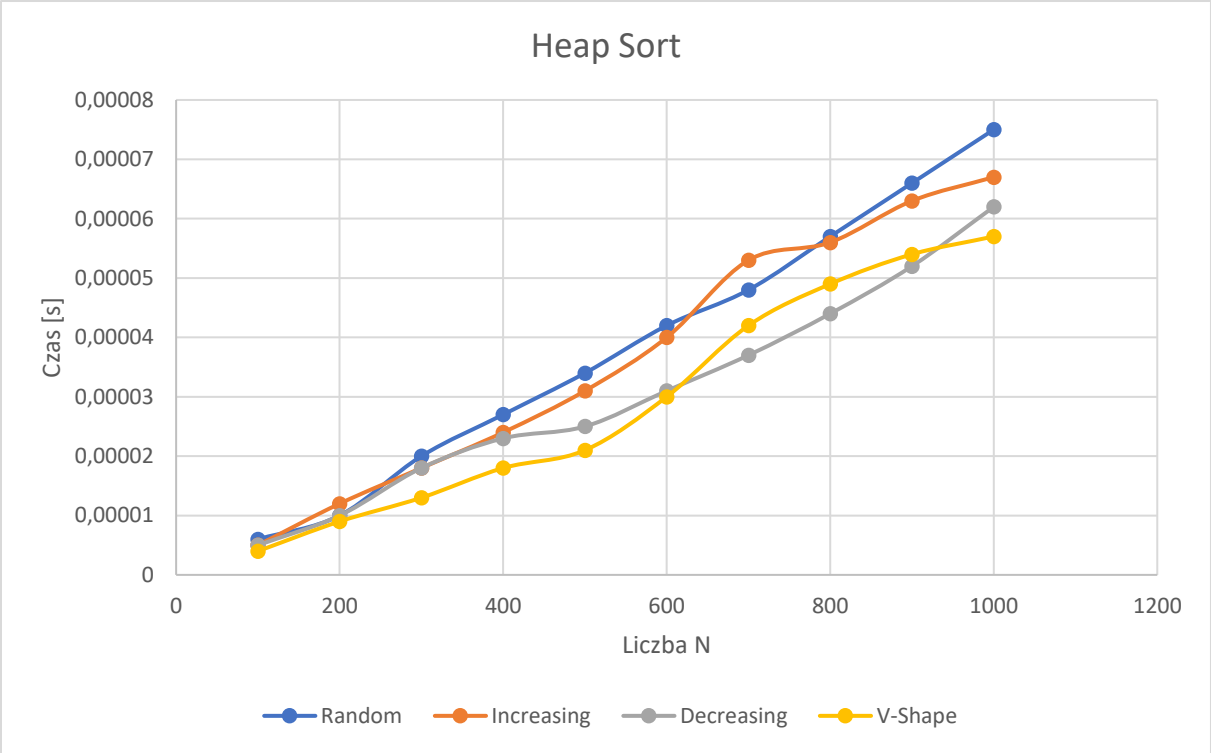
Algorytm ten bardzo szybko może potwierdzić, czy dane które sortujemy są już posortowane, malejąco lub rosnąco w zależności od sposobu w jaki algorytm układa liczby, ponieważ najlepszy przypadek złożoności obliczeniowej wynosi O(n). Pesymistyczna złożoność obliczeniowa osiągana jest, gdy dane uporządkowane są w odwrotny sposób od sposobu w jaki sortuje algorytm, np. gdy dane ustawione są malejąco, gdy algorytm ma je poukładać rosnąco. Wykres przeprowadzonego testu pokazuje wyżej wymienione zależności badanego algorytmu.



Złożoność obliczeniowa:

Przypadek		
Najlepszy	Średni	Najgorszy
$O(nlogn)$	$O(nlogn)$	$O(n^2)$

Sortowanie Quick Sort świetnie nadaje się do stosowania dla nieuporządkowanych danych wejściowych. Pesymistyczną złożoność obliczeniową algorytm posiada dla przesortowanych danych, ponieważ dzieli on zbiór liczb na mniejsze podzbiory i im różniej poukładane liczby tym na równiejsze podzbiory je dzieli. Dla przesortowanych częściowo liczb tworzone podzbiory mają znacząco odbiegającą od siebie liczbę elementów przez co zwiększa się liczba iteracji wykonywanych podczas realizacji algorytmu.



Złożoność obliczeniowa:

Przypadek		
Najlepszy	Średni	Najgorszy
$O(nlogn)$	$O(nlogn)$	$O(nlogn)$

Jest to jeden z lepszych algorytmów do wyboru, kiedy nie mamy pojęcia jak prawdopodobnie wygląda wstępne ułożenie elementów wejściowych. Podobnie jak Selection Sort osiąga on identyczną złożoność obliczeniową do wszystkich sposobów rozmieszczenia danych wejściowych jednak jest ona o wiele niższa i lepiej stosować ten algorytm.

3. **Wnioski:**

Analiza wykresów wydajności algorytmów sortowania pokazuje, że nie ma uniwersalnego algorytmu sortowania danych będący zawsze najszybszym rozwiązaniem, a wybór optymalnego polega na wstępnym przewidzeniu ułożenia danych wejściowych. Są jednak algorytmy, których jedynym plusem jest łatwość implementacji, jednak należy unikać ich używania ze względu na dużą złożoność obliczeniową.