Sprawozdanie

Marcin Prusinowski

Grupa dziekańska: 3a

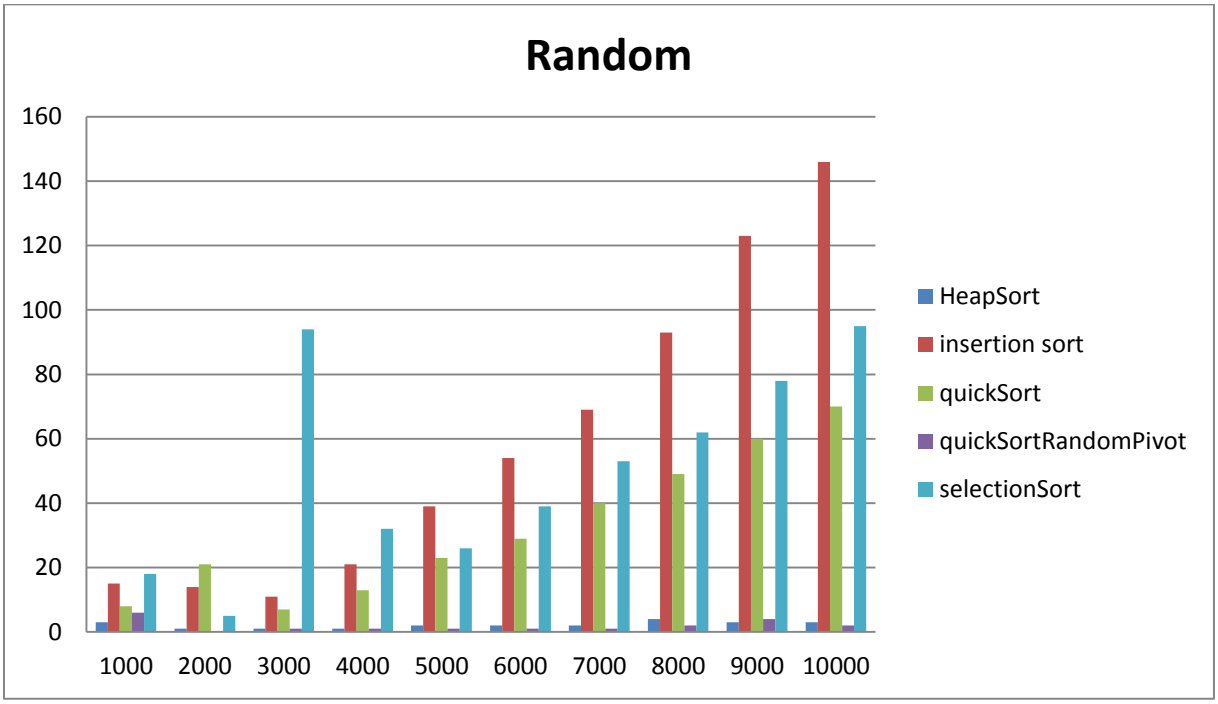
Nr. Albumu: 146400

Algorytmy i Struktury Danych

1.Opis

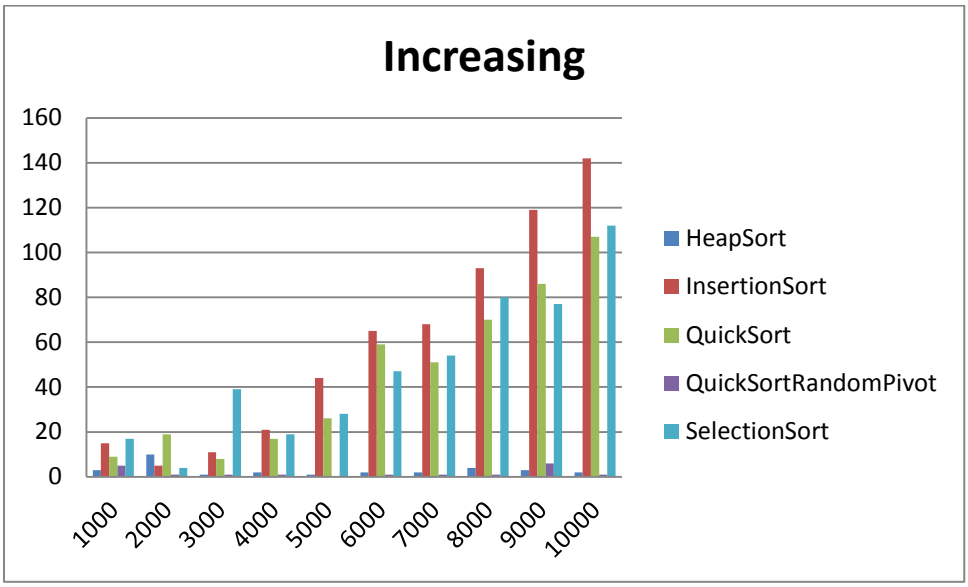
Każdy z algorytmów został poddany pomiarom czasowym wyrażonych w milisekundach względem wielkości tablicy od 1000 do 10000 elementów oraz rodzaju wypełnienia tablicy danymi.

2.Pomiar czasowy dla wartości losowych.



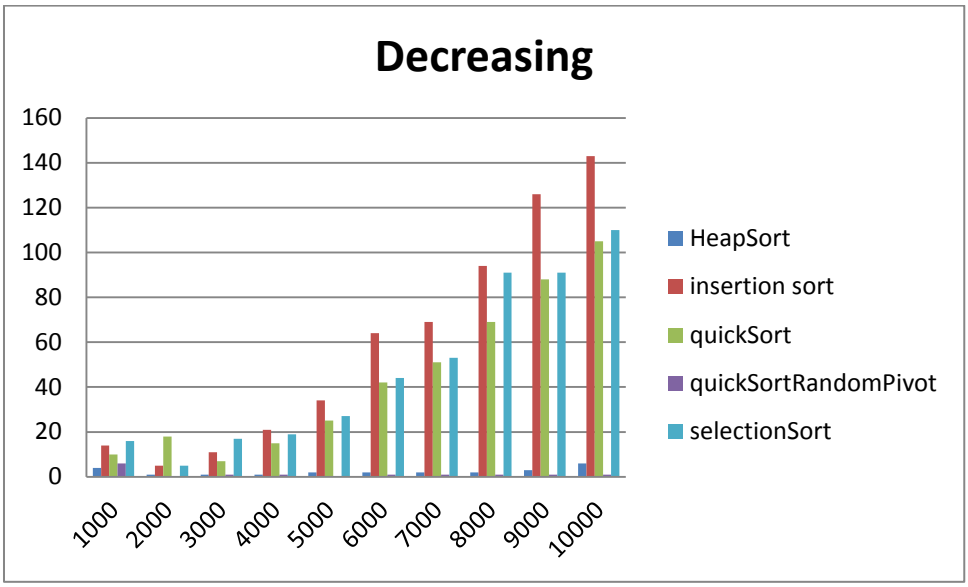
	HeapSort	insertion sort	quickSort	quickSortRandomPivot	selectionSort
1000	3	15	8	6	18
2000	1	14	21	0	5
3000	1	11	7	1	94
4000	1	21	13	1	32
5000	2	39	23	1	26
6000	2	54	29	1	39
7000	2	69	40	1	53
8000	4	93	49	2	62
9000	3	123	60	4	78
10000	3	146	70	2	95

3.Pomiar czasowy dla wartości rosnących



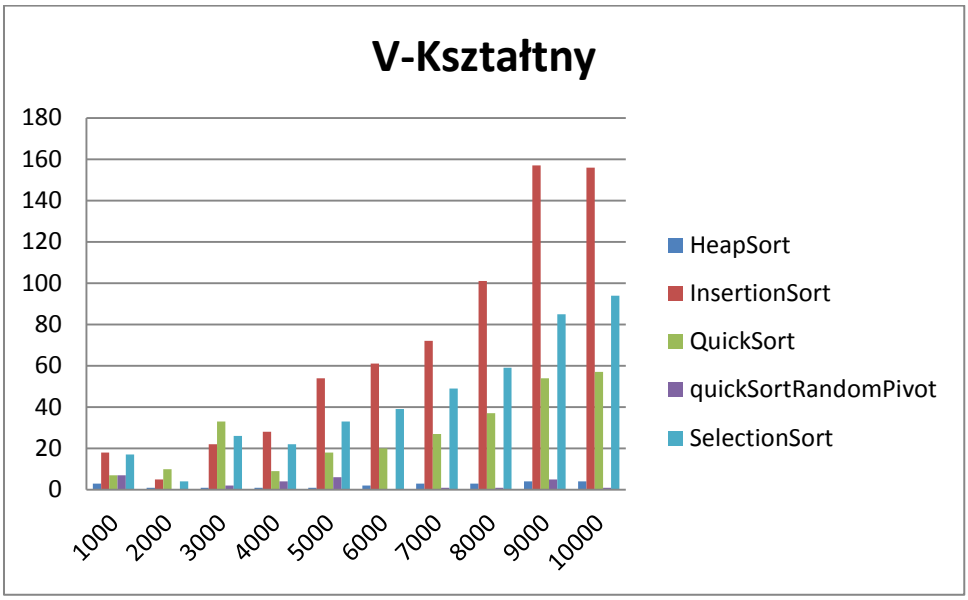
	HeapSort	InsertionSort	QuickSort	QuickSortRandomPivot	SelectionSort
1000	3	15	9	5	17
2000	10	5	19	1	4
3000	1	11	8	1	39
4000	2	21	17	1	19
5000	1	44	26	0	28
6000	2	65	59	1	47
7000	2	68	51	1	54
8000	4	93	70	1	80
9000	3	119	86	6	77
10000	2	142	107	1	112

4.Pomiar czasowy dla wartości malejących



	HeapSort	insertion sort	quickSort	quickSortRandomPivot	selectionSort
1000	4	14	10	6	16
2000	1	5	18	0	5
3000	1	11	7	1	17
4000	1	21	15	1	19
5000	2	34	25	0	27
6000	2	64	42	1	44
7000	2	69	51	1	53
8000	2	94	69	1	91
9000	3	126	88	1	91
10000	6	143	105	1	110

5.Pomiar czasowy dla wartości V-Kształtnych



Wielkość tablicy	HeapSort	InsertionSort	QuickSort	quickSortRandomPivot	SelectionSort
1000	3	18	7	7	17
2000	1	5	10	0	4
3000	1	22	33	2	26
4000	1	28	9	4	22
5000	1	54	18	6	33
6000	2	61	20	0	39
7000	3	72	27	1	49
8000	3	101	37	1	59
9000	4	157	54	5	85
10000	4	156	57	1	94

6.Złożoności Obliczeniowe w najgorszym przypadku dla każdego z algorytmów.

HEAP SORT	O(nlogn)
INSERTION SORT	O(n^2)
QUICK SORT	O(n^2)
QUICK SORT RANDOM PIVOT	O(n^2)
SELECTION SORT	O(n^2)

7.Podsumowanie

Selection sort - Rozkład danych nie ma znaczenia. Algorytm wykonuje taka samą liczbę porównań za każdym razem. Zatem optymistyczny i pesymistyczny przypadek mają złożoność kwadratową.

Insertion sort- Jako iż Algorytm polega na przesuwaniu najmniejszego elementu na początek tablicy, posiadając rosnący rozkład danych działa bardzo szybko wymaga to n-1 porównań czyli jest to złożoność liniowa. Jest to jego najlepszy przypadek. Najgorszym przypadkiem będzie oczywiście malejący rozkład danych gdzie złożoność jest kwadratowa i operacji porównania jest najwięcej.

Quick Sort - Szybkość tego algorytmu zależy od równych podziałów tablicy czyli kolejnych to wywołań rekurencyjnych. W najlepszym przypadku gdzie podziały są równe lub różnią się o 1, złożoność jest logarytmiczna. W najgorszym przypadku gdzie wartości rosną posiada złożoność kwadratową.

Quick Sort Random Pivot - W tym przypadku wiele zależy od losowego pivota, ponieważ każdy podział jakiego dokona jest przypadkowy i raz może wylosować pivot idealnie a dla kolejnego podziału najgorzej. Przy najlepszym przypadku złożoność jest logarytmiczna a przy najgorszym kwadratowa. Pomimo losowości pivota algorytm podczas moich badań okazał się jednym z najszybszych wraz z Heap Sortem.

Heap Sort - Złożoność obliczeniowa jest logarytmiczna dla najgorszego przypadku czyli posortowania odwrotnego, każde wstawienie nowego elementu na stos może wymagać tyle przesunięć ile jest węzłów.

Dla najlepszego przypadku gdzie rozkład jest rosnący złożoność jest również logarytmiczna. Algorytm ten podczas moich badań okazał się jednym z najszybszych wraz z Quick Sort Random Pivot.