# ALGORYTMY

## SORTOWANIA




***Wykonanie***

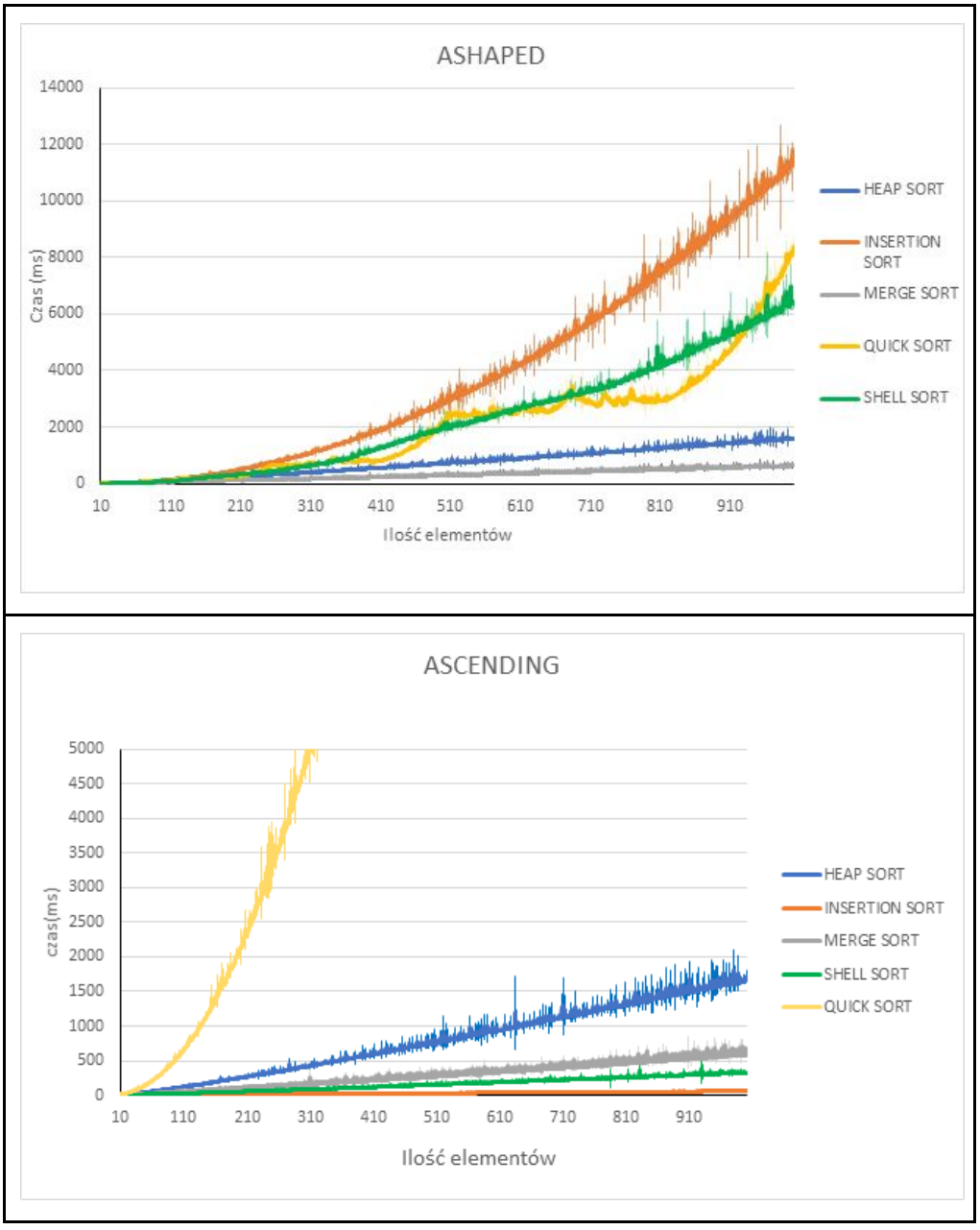
***Adrian Madajewski*** 145406

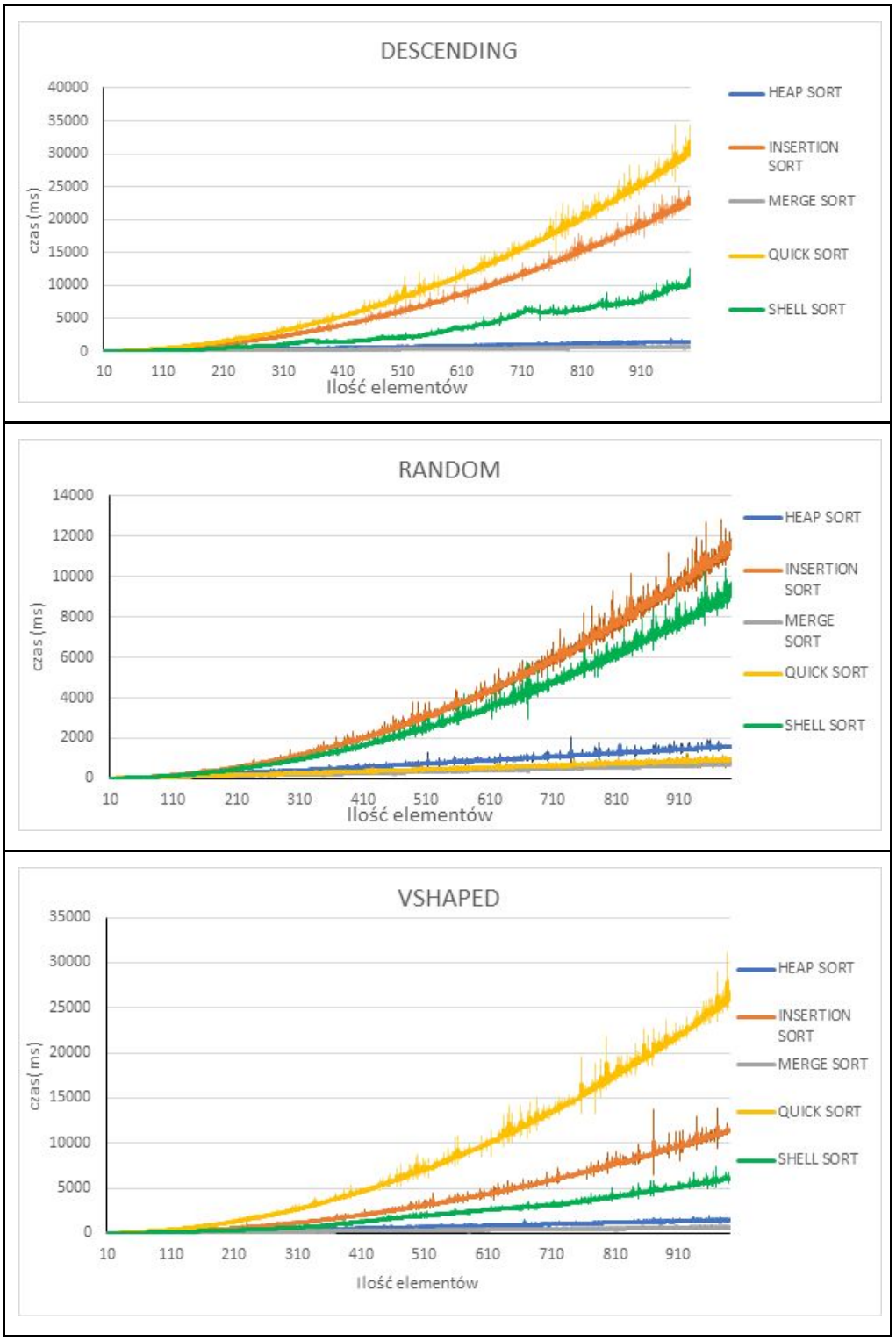
***Michał Kwarta*** 145192

# 1. Wykresy zależności algorytmów od zbiorów danych i operacji

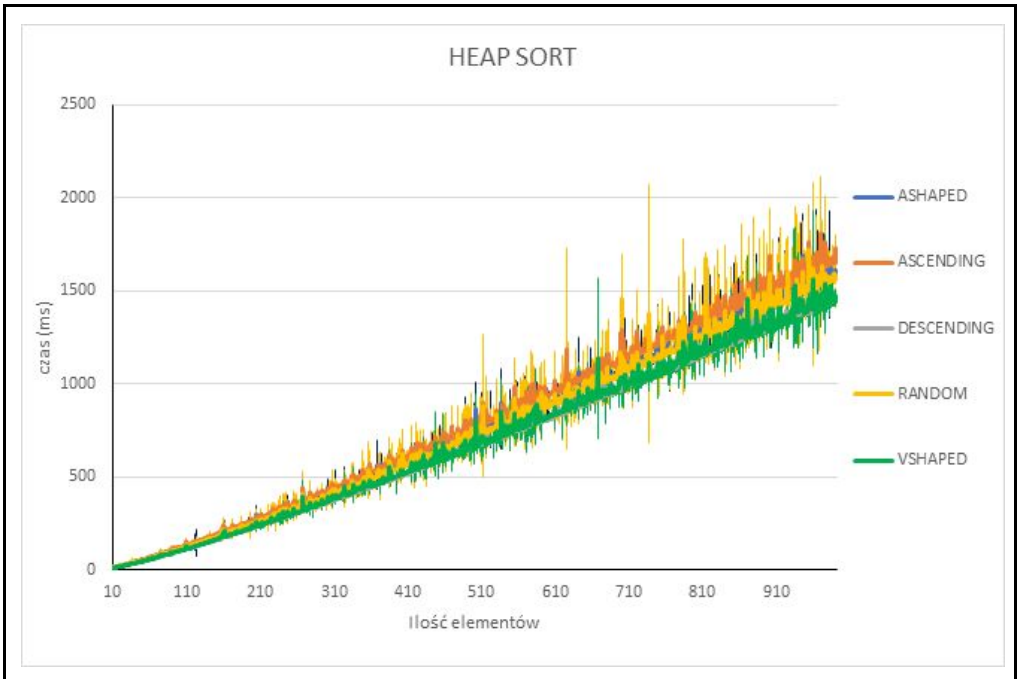
Prezentowane dane są wynikiem symulacji dla każdego zbioru danych: rosnących, malejących, a-kształtnych, v-kształtnych i losowych. Porównywany czas algorytmów jest wyrażony w mikrosekundach, a zakres wielkości danych wynosił od $10$ elementów do $1000$ elementów. Dostęp do arkusza z danymi znajduje się na końcu dokumentu.

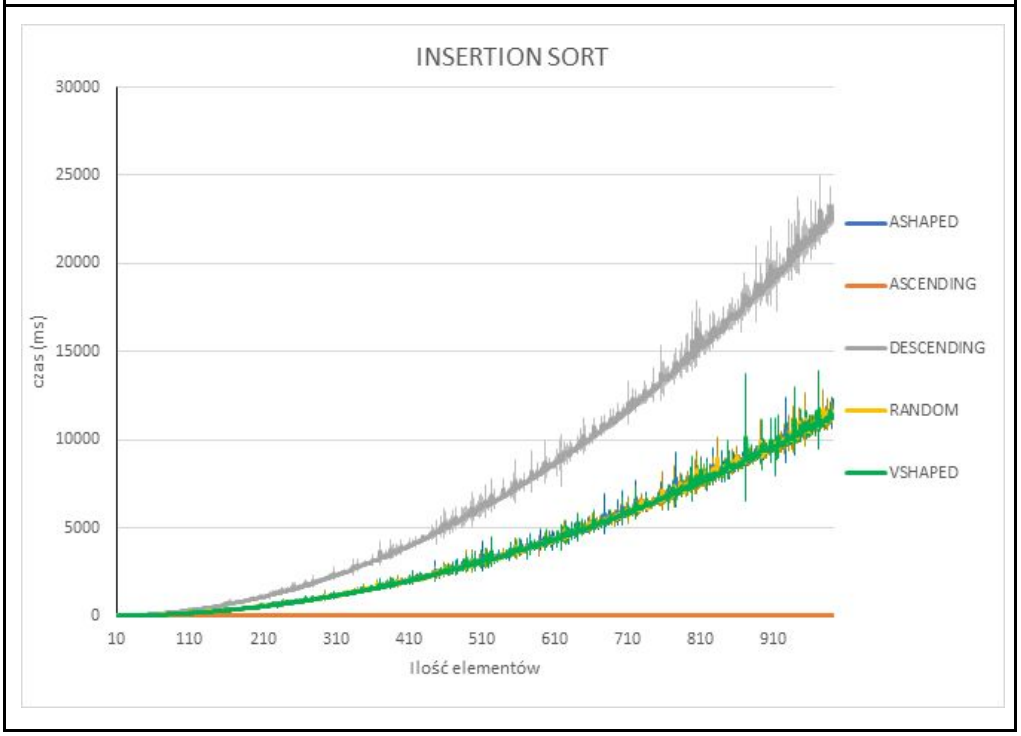
## 1.1. Według rozkładu danych

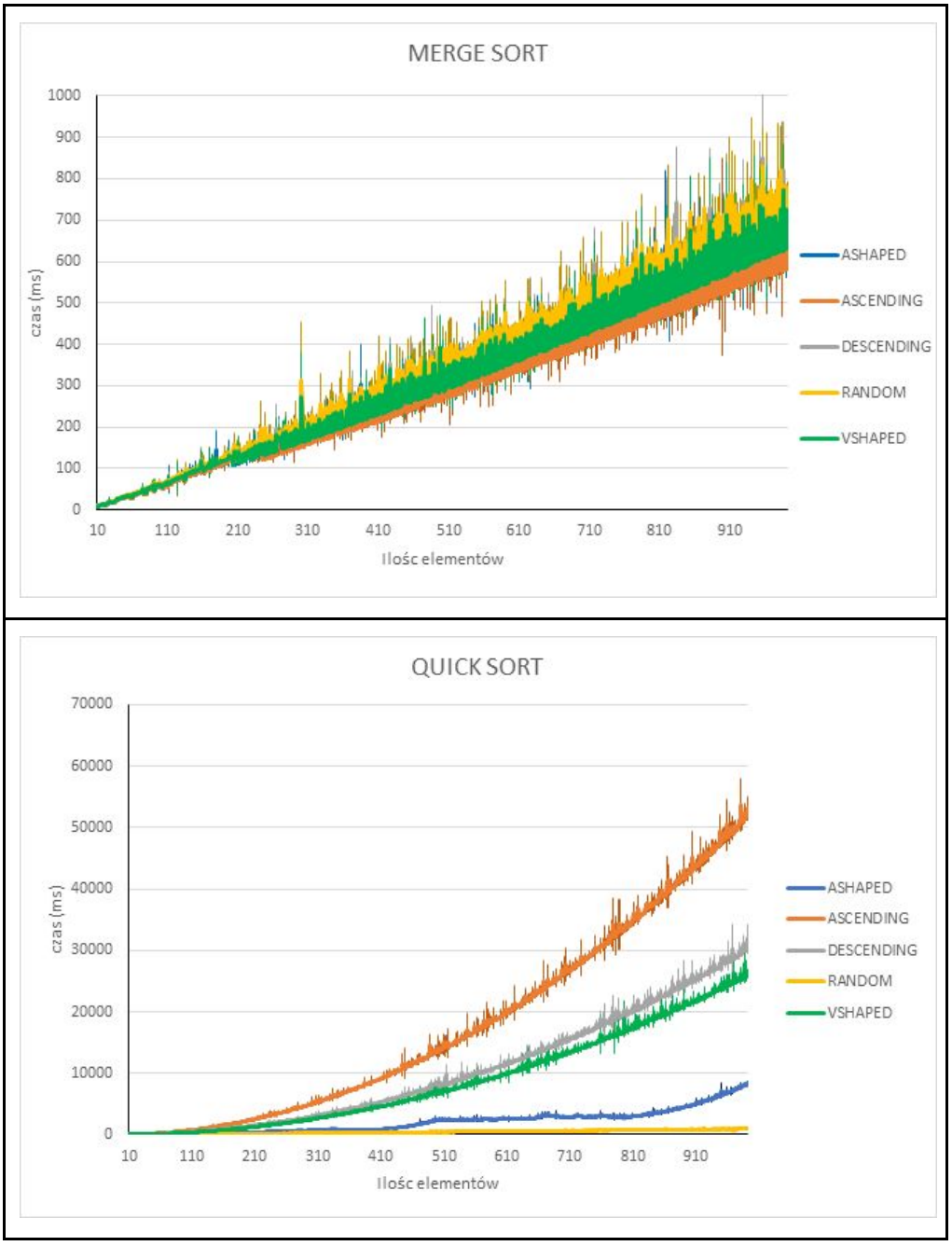


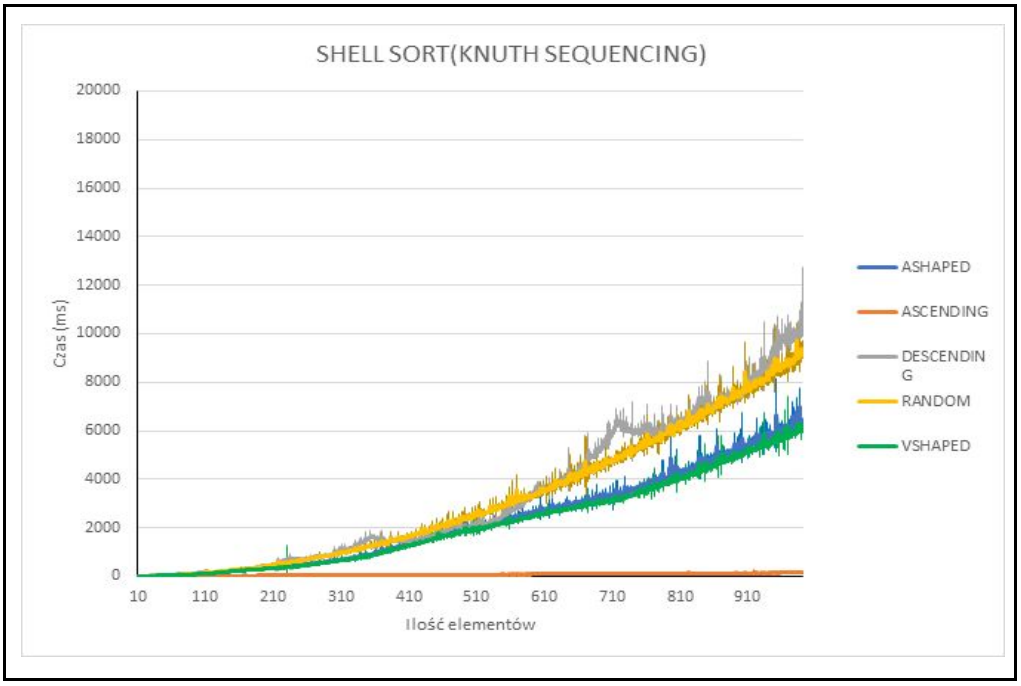

DESCENDING
40000
35000
30000
25000
20000
15000
10000
5000
0
czas (ms)
10
110
210
310
410
510
610
710
810
910
Ilość elementów
HEAP SORT
INSERTION SORT
MERGE SORT
QUICK SORT
SHELL SORT
RANDOM
14000
12000
10000
8000
6000
4000
2000
0
czas (ms)
10
110
210
310
410
510
610
710
810
910
Ilość elementów
HEAP SORT
INSERTION SORT
MERGE SORT
QUICK SORT
SHELL SORT
VSHAPED
35000
30000
25000
20000
15000
10000
5000
0
czas( ms)
10
110
210
310
410
510
610
710
810
910
Ilość elementów
HEAP SORT
INSERTION SORT
MERGE SORT
QUICK SORT
SHELL SORT

## 1.2. Według czasów obliczeń

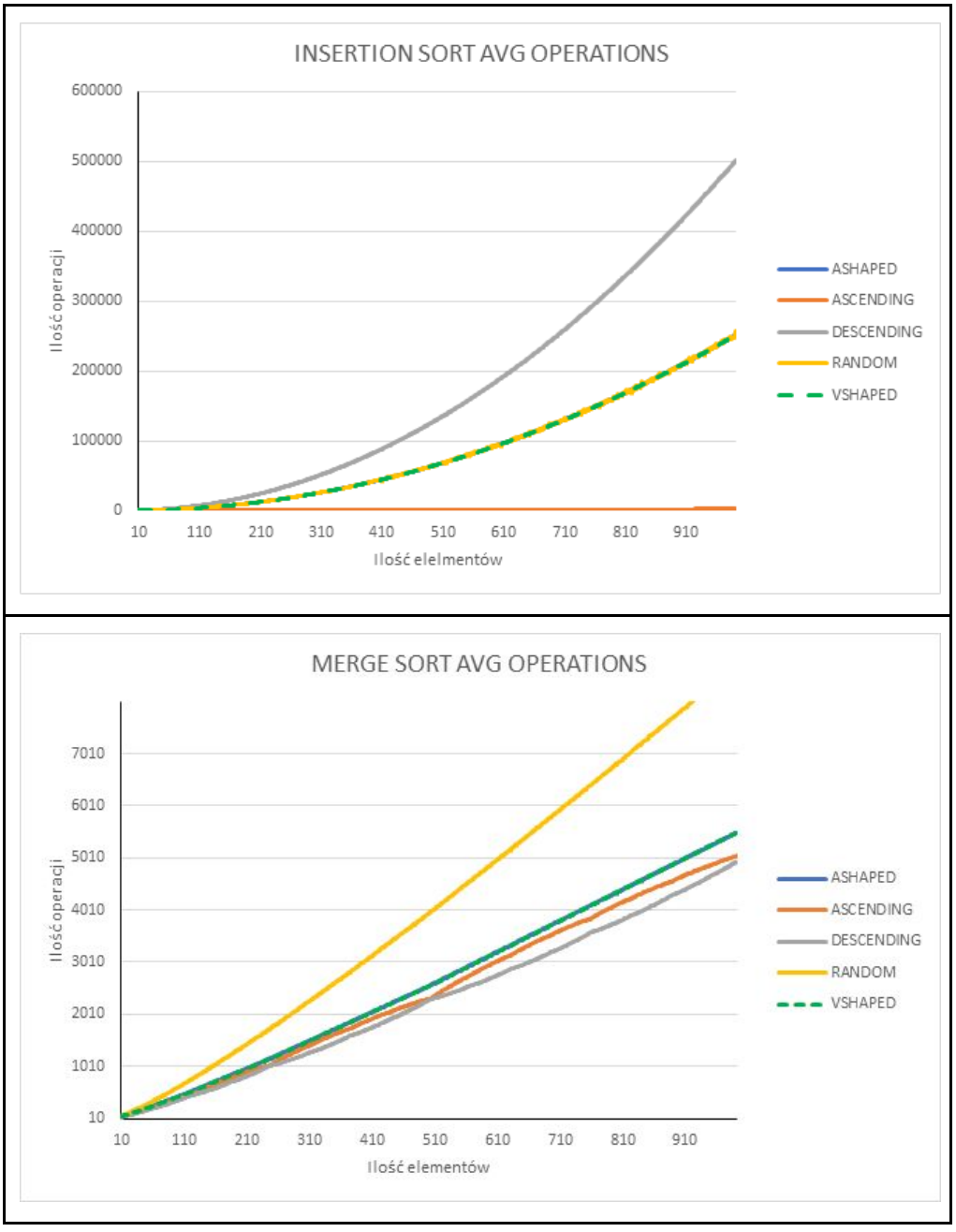


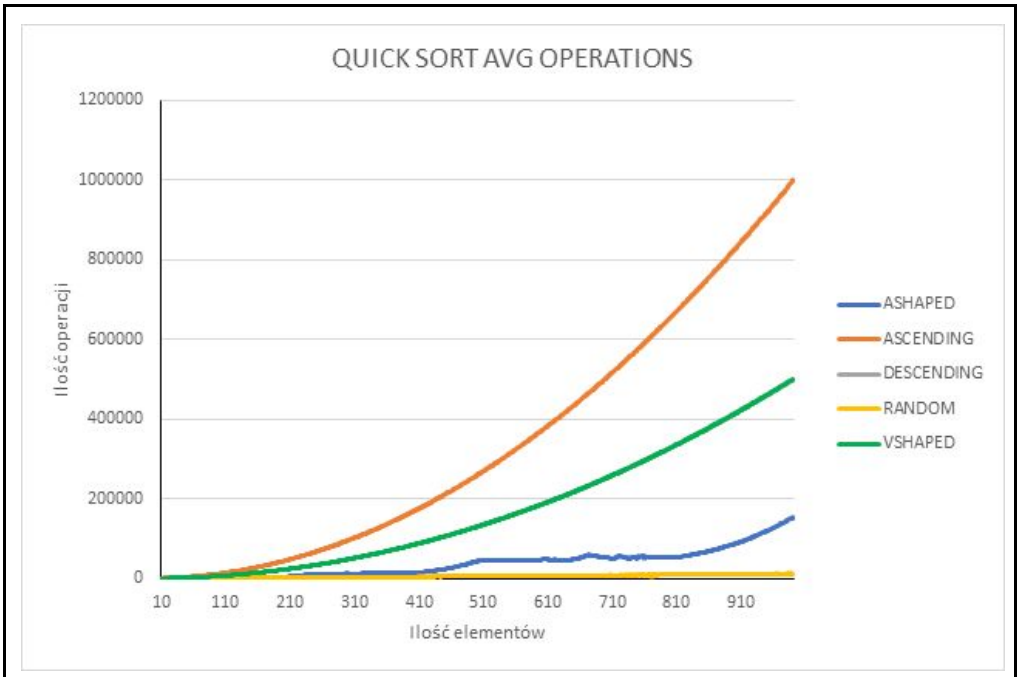

MERGE SORT
1000
900
800
700
600
500
400
300
200
100
0
czas (ms)
10
110
210
310
410
510
610
710
810
910
Ilośc elementów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED
QUICK SORT
70000
60000
50000
40000
30000
20000
10000
0
czas (ms)
10
110
210
310
410
510
610
710
810
910
Ilośc elementów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED

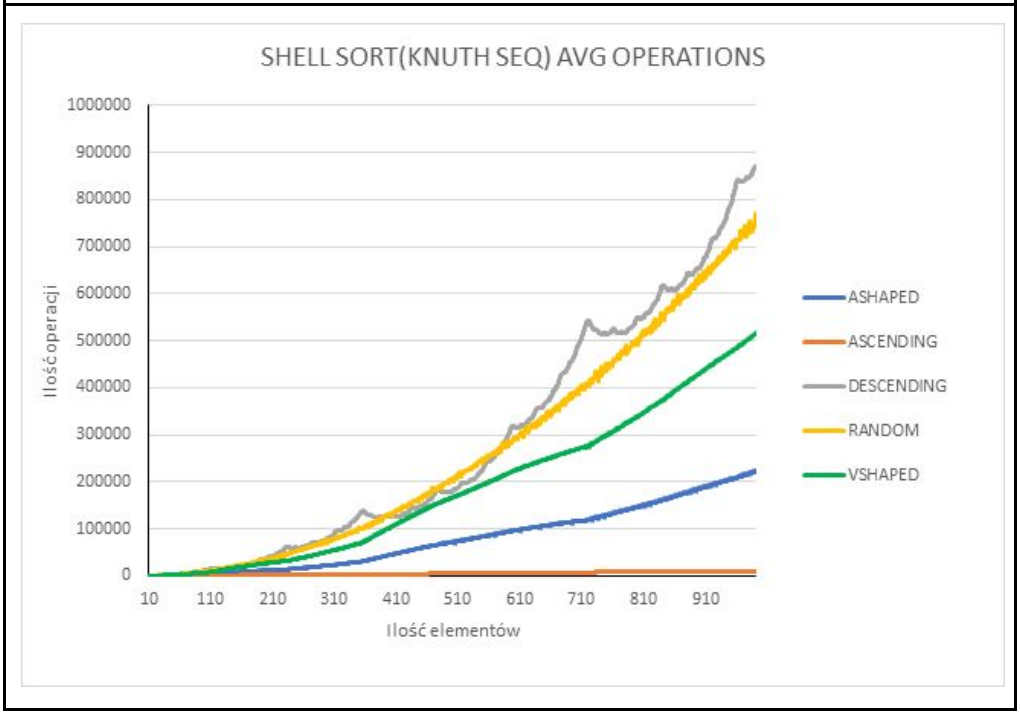



## 1.3. Według średniej ilości wykonywanych operacji


INSERTION SORT AVG OPERATIONS
600000
500000
400000
300000
200000
100000
0
Ilość operacji
10
110
210
310
410
510
610
710
810
910
Ilość elelmentów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED


MERGE SORT AVG OPERATIONS
7010
6010
5010
4010
3010
2010
1010
10
Ilość operacji
10
110
210
310
410
510
610
710
810
910
Ilość elementów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED


QUICK SORT AVG OPERATIONS
1200000
1000000
800000
600000
400000
200000
0
Ilość operacji
10
110
210
310
410
510
610
710
810
910
Ilość elementów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED


SHELL SORT(KNUTH SEQ) AVG OPERATIONS
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
Ilość operacji
10
110
210
310
410
510
610
710
810
910
Ilość elementów
ASHAPED
ASCENDING
DESCENDING
RANDOM
VSHAPED

# 2. Tabela złożoności obliczeniowej:

| Algorytm | Złożoność obliczeniowa | |
|---|---|---|
| **Insertion Sort** | optymistyczna | $O(n)$ |
| | średnia | $O(n^2)$ |
| | pesymistyczna | $O(n^2)$ |
| **Merge Sort** | optymistyczna | $O(n \cdot logn)$ |
| | średnia | $O(n \cdot logn)$ |
| | pesymistyczna | $O(n \cdot logn)$ |
| **Heap Sort** | optymistyczna | $O(n \cdot logn)$ |
| | średnia | $O(n \cdot logn)$ |
| | pesymistyczna | $O(n \cdot logn)$ |
| **Shell Sort**<br>przyrosty Knutha | optymistyczna | $O(n^{1,14})$ |
| | średnia | $O(n^{1,14})$ |
| | pesymistyczna | $O(n^{1,15})$ |
| **Quick Sort**<br>pivot - ostatni element | optymistyczna | $O(n \cdot logn)$ |
| | średnia | $O(n \cdot logn)$ |
| | pesymistyczna | $O(n^2)$ |

# 3. Wnioski

## 3.1. Insertion Sort

Złożoność optymistyczna występuje tylko dla elementów już uporządkowanych (ascending), natomiast taka sytuacją zdarza się prawie nigdy co czyni algorytm mało wydajnym w porównaniu do innych metod. Ze względu na złożoność typową tego algorytmu - $O(n^2)$ - w przypadku średnim i pesymistycznym - gdzie w tym ostatnim zbiór jest posortowany odwrotnie. Z wykresów zauważamy też że jest to najmniej wydajny algorytm sortowania. Przeprowadzone testy potwierdzają zależność operacji porównania i zamiany elementów do przedstawionych na wykresach czasów wykonywania.

## 3.2. Merge Sort

Implementacja MergeSorta czyni go jednym z najbardziej odpornym na dane algorytmem sortowania. Z wykresów widzimy, że algorytm wykonuje się w prawie identycznych czasach bez względu na podane dane. Ilość operacji porównania również jest niemalże taka sama, z wyjątkiem danych losowych, gdzie wykres przewyższa każdy inny. Wynika to z dużej losowości dzielonych zbiorów na podzbiory losowe. Sam algorytm jest jednym z najwydajniejszych algorytmów sortowania w porównaniu do reszty. Przeprowadzone testy potwierdzają zgodność szybkości wykonania oraz ilość przeprowadzanych operacji z jego złożonością.

## 3.3. Heap Sort

Algorytm HeapSort jest również jednym z najszybszych algorytmów sortowania ze względu na pracę in situ (w miejscu), przez co nie wymaga tworzenia dodatkowego zbioru danych. Składa się z dwóch etapów - walidacji kopca, która jest jedynie zależna od wielkości zbioru danych oraz samego algorytmu sortowania - przez tworzenia kopca rosnącego. Wynika z tego, że tak samo jak w przypadku Mergesorta algorytm jest niezależny od danych wejściowych - jednak dla niektórych wykonuje się wolniej od chociażby QuickSorta, natomiast przewyższa go w przypadku pesymistycznym.

### 3.4. **ShellSort** (z przyrostami Knutha)

ShellSort jest pewną modyfikacją sortowania przez wstawianie. Idea działania algorytmu polega na sortowaniu najpierw elementów leżących daleko od siebie dążąc do coraz to lepszego uporządkowania. Zasadniczo jest wydajniejszy niż algorytm InsertionSort wykonując dodatkowo mniej operacji. Jest algorytmem zależnym od implementacji ustalonych przyrostów - dla innego ich dobrania możemy otrzymać zdecydowanie lepsze rezultaty.

### 3.5. **QuickSort** (pivot = ostatni element ciągu)

Algorytm QuickSort jest wydajnym algorytmem sortowania z wyjątkiem przypadku pesymistycznego gdzie dane są posortowane malejąco - skutkuje to tym, że w bieżącej implementacji podział rekurencyjny danych jest niekorzystny, ponieważ dzieli się zawsze na jednoelementowy  i resztę. Słabo radzi sobie też z danymi V-kształtnymi ze względu na dużą liczbę przeprowadzanych podziałów i niekorzystnym rozstawieniu elementów po wybraniu pivota.

# 4. Informacje dodatkowe

*strona github z kodami źródłowymi*
*link do arkusza z danymi*

**Platforma testowa:**

| | |
|---|---|
| procesor: | Intel i7 7700k 4.2 GHz |
| ram: | 8 GB RAM DDR4 CL16 |
| karta graficzna: | geforce gtx 1060 6gb |
| system operacyjny: | windows 7 64-bit |
| płyta główna: | MSI Z270-A PRO |
| dysk: | HDD 1000 GB |