**Algorytmy sortowania – sprawozdanie №1**

Andrei Staravoitau, nr. albumu: 150218, grupa: I6, rok: I, sem: 2

2 kwietnia 2021

1. **Wstęp**

Sprawozdanie stanowi analizę wybranych algorytmów sortowania.

Do posortowania wybrano tablice o długości od 10 do 7500 elementów. Każdy punkt na wykresie jest średnią z 8 pomiarów wykonanych dla danej wielkości tablicy.

Wykresy prezentujące zależność liczby operacji od liczby elementów w tablicy prezentują zależność sumy operacji porównań i przestawień elementów wykonywanych podczas sortowania.

Testy zostały wykonane z użyciem systemu operacyjnego Windows 10 (64-bit).

Język programowania w którym zaimplementowano algorytmy: Python

Procesor: Intel Core i7-3610 @ 2.30Ghz 2.30 GHz

RAM: 8GB

1. **Code**

Code znajduje się w załączniku “Sprawodzanie1.py”

Dla Heap sortu zostały zrobione dwie funkcje: heapSort, heapify.

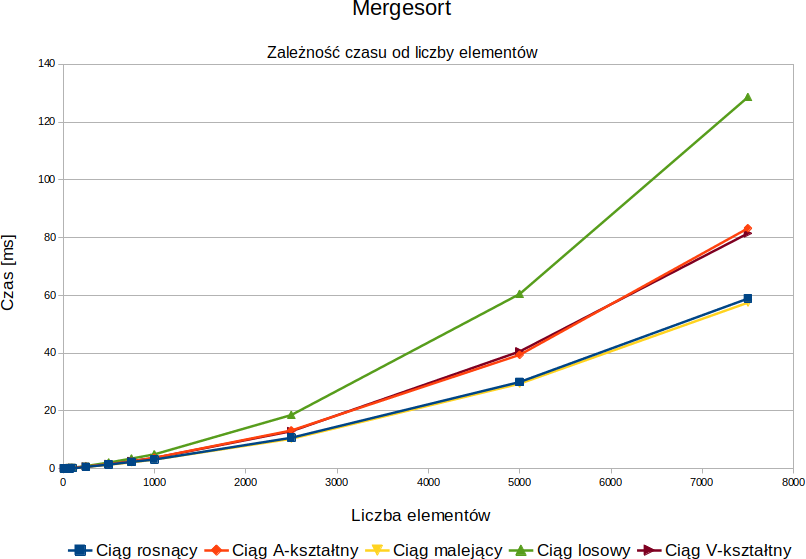
Dla Shell sortu zostały zrobione trzy funkcje: shell\_sort, ins\_sort, knuth\_przyrosty.

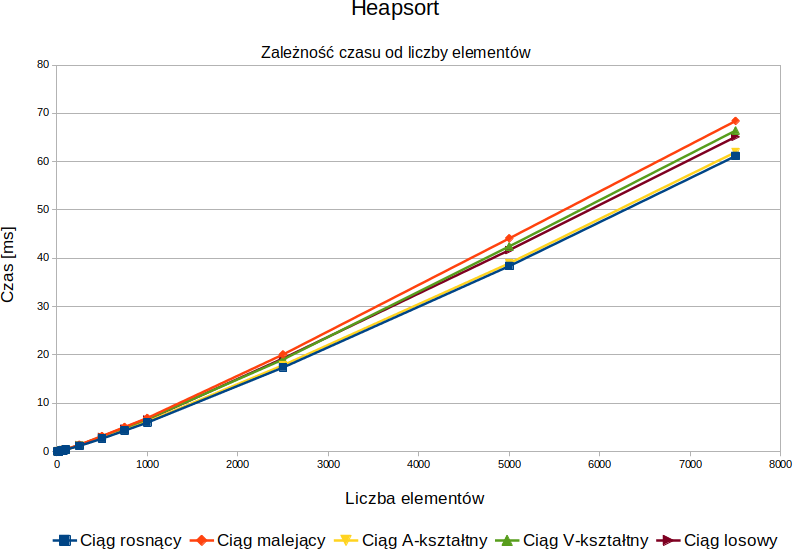
Pozostałe algorymy sortowania mają jedną funkcję z odpowiednymi nazwami.

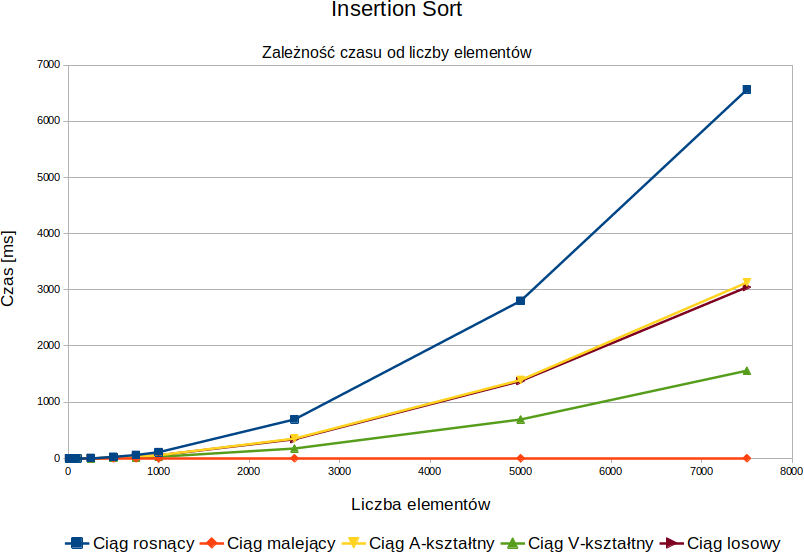
Na początku użytkownik musi podać dowolny ciąg liczb, po tym program tworzy drugi 10-elementowy ciąg z losowymi liczbami.

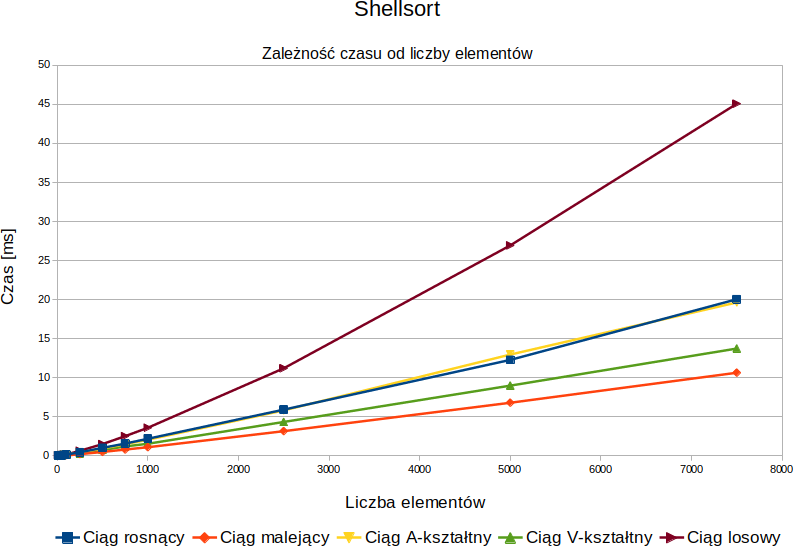
Na wyjściu użytkownik otrzymuje informację zawierającą swój posortowany ciąg, czas wykonania sortowania dwóch ciągów, liczbę porównań i przestawień, wykonanych algorytmem sortowania oraz dodatkową informację o niektórych algorytmach.

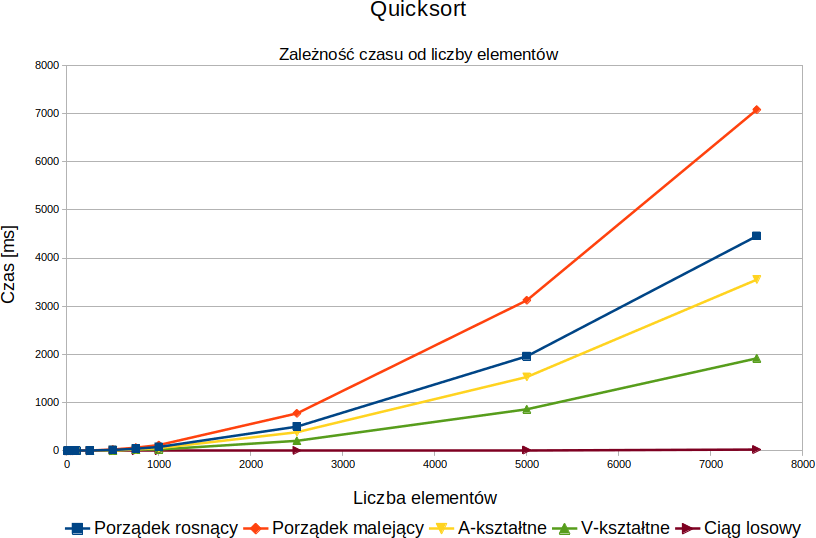
1. **Porównanie zależności czasu od liczby elementów w tablicy**



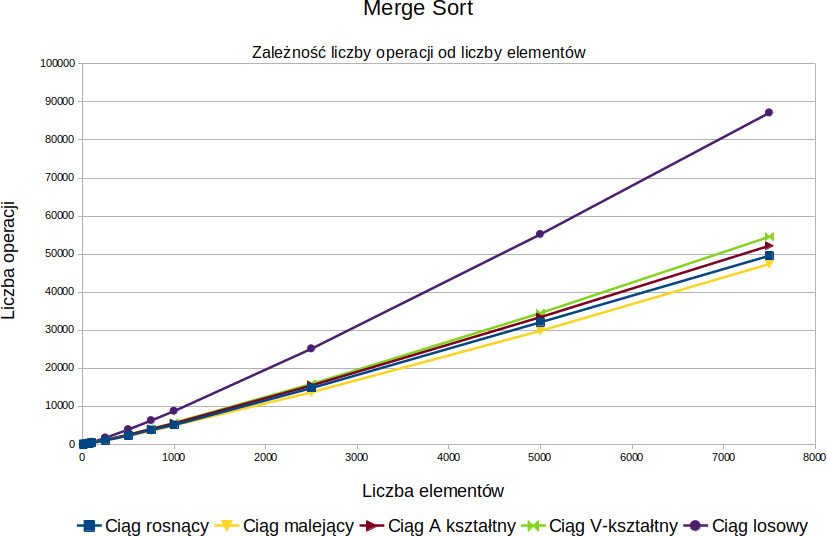


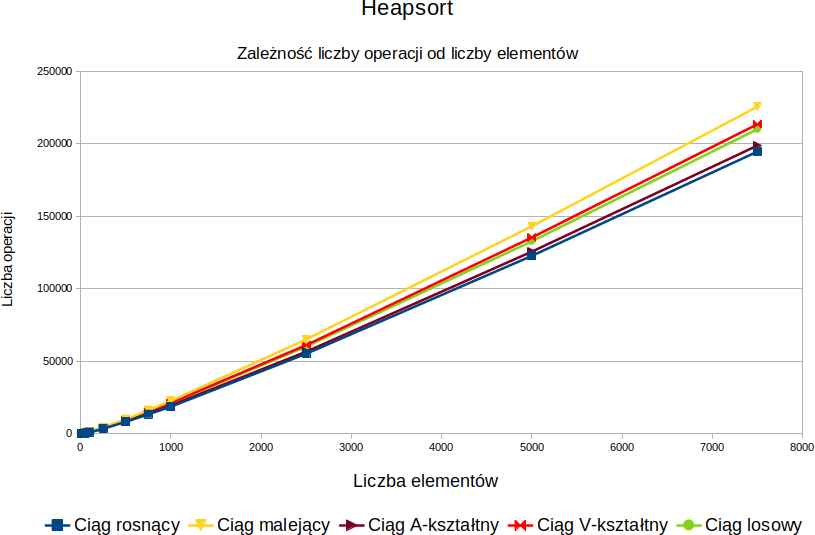


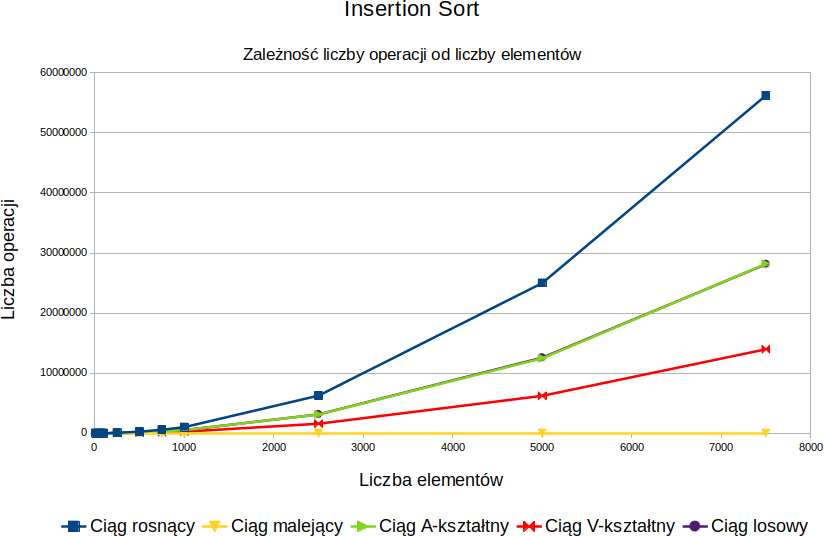


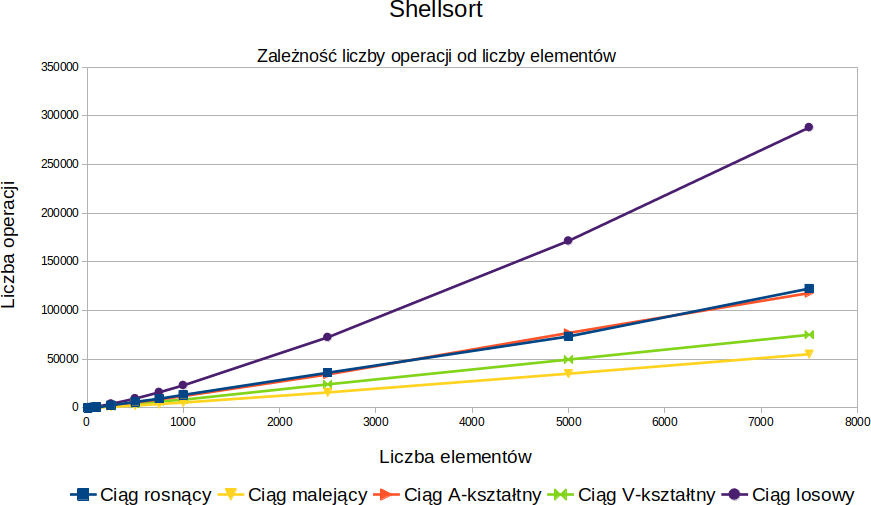


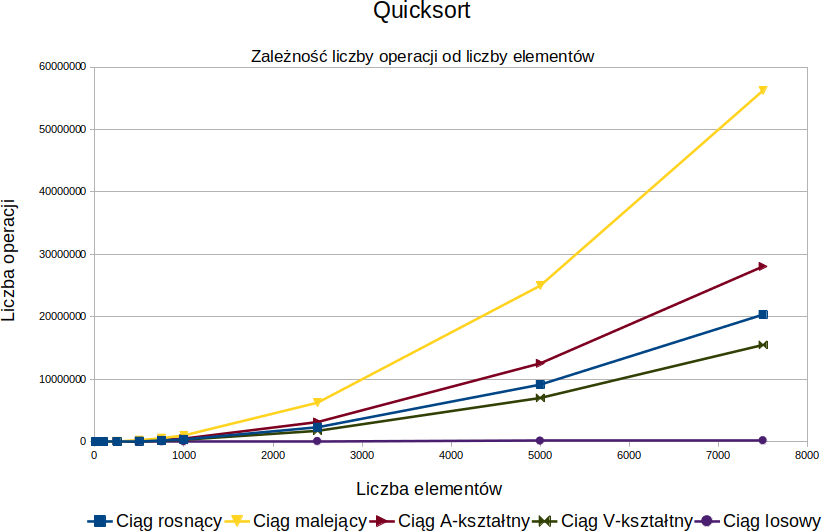
1. **Porównanie zależności liczby operacji od liczby elementów w tablicy**

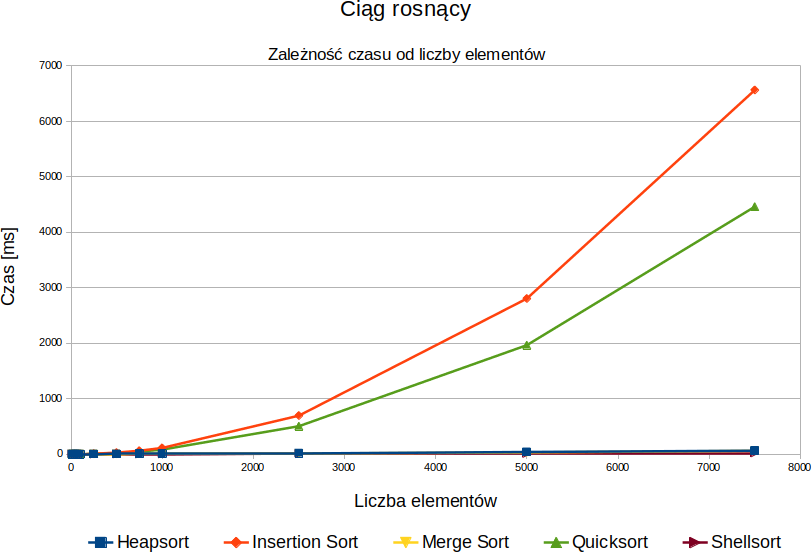


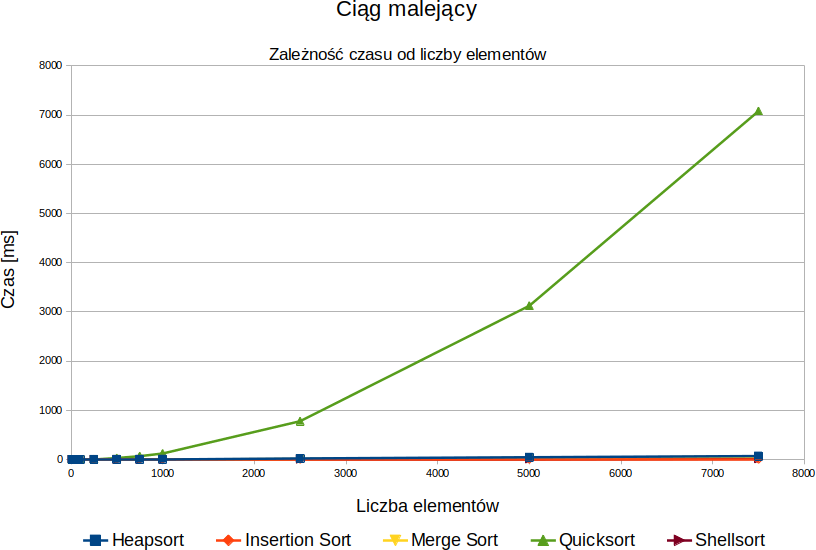


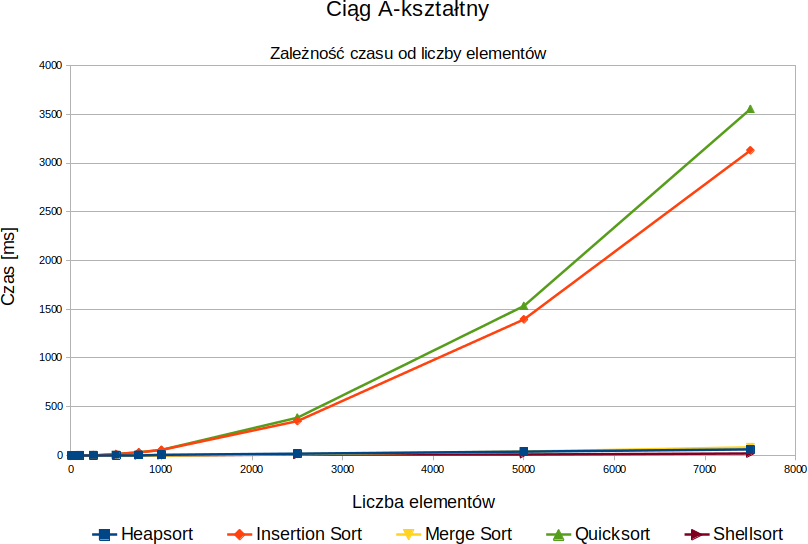


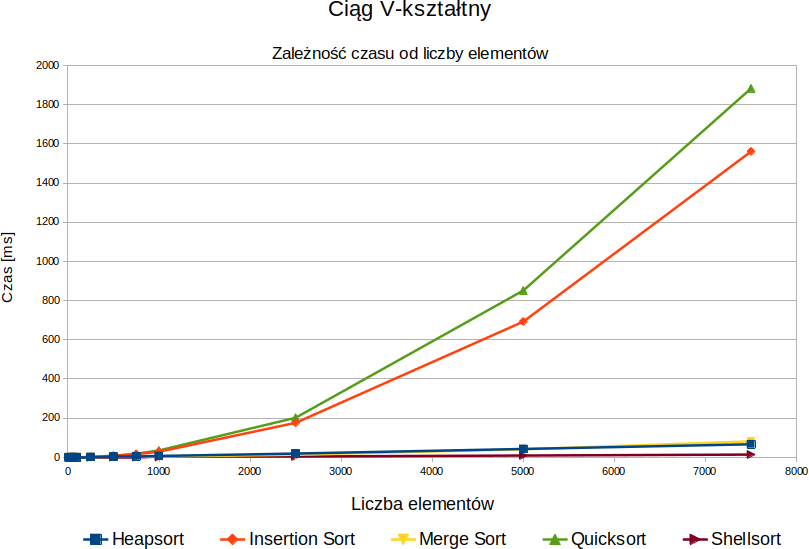


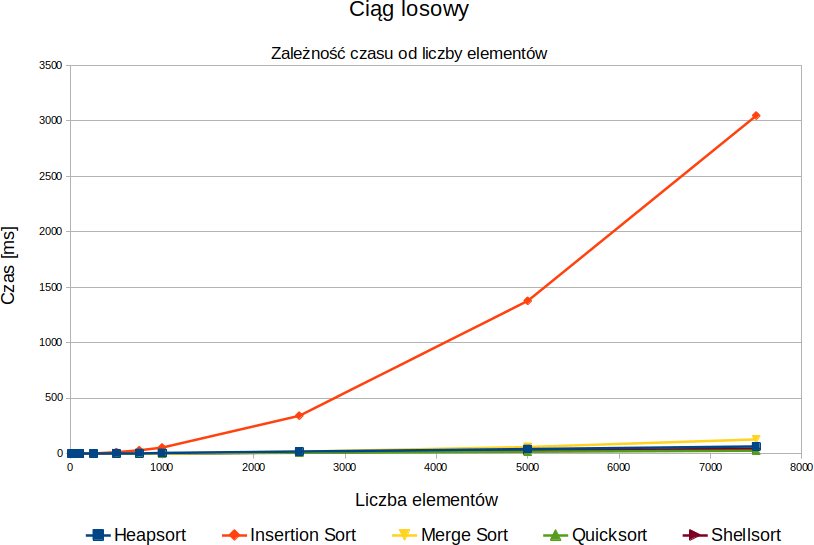


1.  **Porównanie zależności czasu od liczby elementów w tablicy dla różnych typów danych wejściowych**









1. **Złożoność obliczeniowa**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorytm** | **Pesymistycznie** | **Średnio** | **Optymistycznie** |
| Merge sort | *O*(*n* log *n*) | *O*(*n* log *n*) | *O*(*n* log *n*) |
| Heap sort | *O*(*n* log *n*) | *O*(*n* log *n*) | *O*(*n* log *n*) |
| Insertion sort | *O*(*n*2) | *O*(*n*2) | *O*(*n*) |
| Shell sort | *O*(*n*2) | *O*(*n* log *n*) | *O*(*n* log *n*) |
| Quick sort | *O*(*n*2) | *O*(*n* log *n*) | *O*(*n* log *n*) |
| Bubble sort | *O*(*n*2) | *O*(*n*2) | *O*(*n*2) |
| Selection sort | *O*(*n*2) | *O*(*n*2) | *O*(*n*2) |

1. **Podsumowanie**

Wynikiem wykonanego sprawozdania jest wyjaśnienie, że na czas pracy algorytmu w znacznym stopniu wpływa sposób uporządkowania danych wejściowych. Liczba wykonanych operacji jest proporcjonalna złożoności obliczeniowej, o czym można wnioskować zgodnie z przedstawionymi wykresami. Czas wykonania obliczeń dodatkowo zależy of mocy obliczeniowej procesora oraz od liczby wykonywanych im procesów.