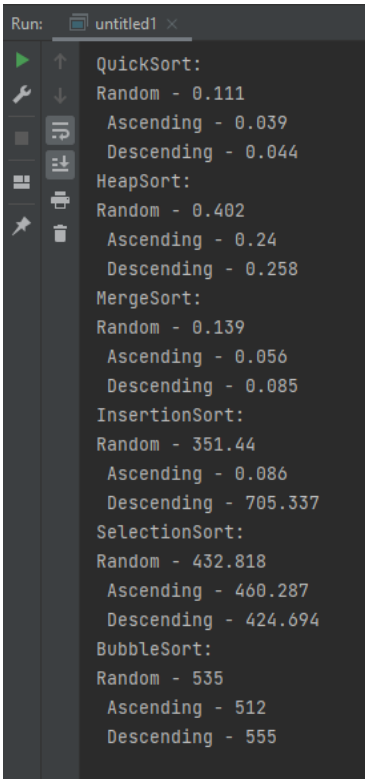
Jakub Krzynowek s22835

Analiza Algorytmów Sortowania

1. Przebieg doświadczenia

Doświadczenie polegało na napisaniu czterech różnych algorytmów sortujących oraz wygenerowania trzech plików tekstowych z danymi wejściowymi potrzebnymi do testowania napisanych programów. Ja z ciekawości, eksperyment, postanowiłem przeprowadzić na sześciu.

Dwa algorytmy które były wymagane to HeapSort oraz QuickSort, jako dodatkowe algorytmy wybrałem BubbleSort, InsertionSort, MergeSort oraz SelectionSort. Wygenerowane pliki tekstowe posiadały równo milion dodatnich liczb naturalnych (typu int) z przedziału od 0 do 10.000.



2. Wyniki testów oraz złożoności.

	Dane losowe	Dane posortowane ros.	Dane posortowane mal.
BubbleSort	535s	512s	555s
InsertionSort	351,44s	0,086s	705,337s
MergeSort	0,139	0,056	0,085
SelectionSort	432.818s	460.287s	424.694s
HeapSort	0,402s	0,24s	0,085s
QuickSort	0,111s	0,039s	0,044s

3. Algorytmy uporządkowane według średniego czasu oraz ich złożoności:

1. QuickSort – (n log(n))
2. MergeSort - (n log(n))
3. HeapSort – (n log(n))
4. InsertionSort - (n^2)
5. SelectionSort - (n^2)
6. BubbleSort - (n^2)

4. Analiza wyników

- QuickSort – Najszybszy algorytm sortujący, idealny przykład zastosowania idei „dziel i zwyciężaj”

- MergeSort – Algorytm korzystający z wyżej wymienionej idei, jest minimalnie wolniejszy od QuickSort

- HeapSort – Mimo swojej unikatowej metody sortowania jest zaskakująco szybki

- InsertionSort – Algorytm którego używamy w prawdziwym życiu np. podczas sortowania talii kart. Ma najkrótszy ale i najdłuższy czas ze wszystkich testów. Jego niestabilność jest zdecydowanie niepożądana.

- SelectionSort – Jest trochę szybszy od BubbleSort ale nadal zdecydowanie wolniejszy od najszybszego

- BubbleSort – Jest najprostszym ale również najwolniejszym algorytmem

5. Wnioski

Zgodnie z przewidywaniami prędkość sortowania jest bezpośrednio związana ze złożonością algorytmu. Zaskoczyła mnie mała rozbieżność między czasami poszczególnych programów o takiej samej złożoności, jak również ogromna różnica między tymi algorytmami które miały inne zbieżności. Doświadczenie uświadomiło mi jak wielką rolę odgrywa optymalizacja kodu w przypadku operowania na dużej ilości danych.