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.	Średni czas
BubbleSort	441s	437s	453s	437s
InsertionSort	243s	0,2s	456s	233s
MergeSort	1,1s	0,5s	0,8s	0,8s
SelectionSort	404s	399s	418s	407s
HeapSort	1,5s	1,9s	1,1s	1,5s
QuickSort	0,2s	0,2s	0,2s	0,2s

3. Algorytmy uporządkowane według średniego czasu oraz ich złożoności:
1. QuickSort – (n log(n))
1. MergeSort - (n log(n))
2. HeapSort – (n log(n))
3. InsertionSort - (n^2)
4. SelectionSort - (n^2)
5. BubbleSort - (n^2)

4. Analiza wyników
- QuickSort – Najszybszy algorytm sortujący, idealny przykład działania idei „dziel i zwyciężaj”

- MergeSort – Algorytm również 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 daleko w tyle za Quicksort

- 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 poszczególnymi programami 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.