Análisis de algoritmos

ShellSort.

|  |  |
| --- | --- |
| Nombre del Algoritmo | ShellSort |
| Mejor caso | El mejor caso se presenta cuando los datos están casi ordenados. |
| Complejidad en el mejor caso | O(N\* log N) |
| Peor caso | El peor caso se presenta cuando los datos están aleatorios. |
| Complejidad en el peor caso | O(N^1.5) |
| Algoritmo inplace | Si |
| Adaptativo | Si |
| Estable | No |

MergeSort.

|  |  |
| --- | --- |
| Nombre del Algoritmo | MergeSort |
| Mejor caso | En general, este ordenamiento se demora lo mismo para todos los casos, ya sea que estén casi ordenados, inversos, aleatorios o con datos repetidos. |
| Complejidad en el mejor caso | O(N\* log N ) |
| Peor caso | En general, este ordenamiento se demora lo mismo para todos los casos, ya sea que estén casi ordenados, inversos, aleatorios o con datos repetidos. |
| Complejidad en el peor caso | O(N \* log N) |
| Algoritmo inplace | No |
| Adaptativo | No |
| Estable | Si |

QuickSort.

|  |  |
| --- | --- |
| Nombre del Algoritmo | QuickSort |
| Mejor caso | El mejor caso se presenta cuando los datos están en reversa. |
| Complejidad en el mejor caso | O(N \* log N) |
| Peor caso | El peor caso se presenta cuando los datos tienen muchos repetidos. |
| Complejidad en el peor caso | O(N^2) |
| Algoritmo inplace | Si |
| Adaptativo | No |
| Estable | No |

Análisis de tiempo según tipo de ordenamiento.

Los datos se analizaron en un portátil con procesador Intel i7 7th generación y 12 Gb de Ram.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ShellSort(mseg) | MergeSort(mseg) | QuickSort(mseg) |
| Tiempo de ejecución 1 | 463 mseg | 267 mseg | 451 mseg |
| Tiempo de ejecución 2 | 349 mseg | 263 mseg | 361 mseg |
| Tiempo de ejecución 3 | 345 mseg | 261 mseg | 349 mseg |
| Tiempo de ejecución promedio (mseg): | 385,66 mseg | 263,66 mseg | 387 mseg |

Conclusión:

Por el tiempo de ejecución, para el caso general, el algoritmo más eficiente es MergeSort. El siguiente algoritmo en eficiencia es QuickSort y el algoritmo menos eficiente fue ShellSort. Pero, aunque MergeSort es el más eficiente en tiempo por la forma que se desarrolla Merge gasta más memoria.

Juan Pablo Sarmiento

Erik Andre Nielsen