**Estructuras de datos: Taller 3**

Luis Alejandro González Diaz // 201622497

Geisson Ponce Solano // 201913908

**Parte 7.**

a y b.

|  |  |
| --- | --- |
| **Nombre del Algoritmo** | ShellSort |
| **Complejidad en el mejor caso (notación O)** | O(n^2) |
| **Peor caso** | depende del orden de los ítems, pero no varía tanto. |
| **Complejidad en el peor caso (notación O)** | O(n^6/5) |
| **Algoritmo Inplace** | si |
| **Algoritmo Adaptativo** | no |
| **Algoritmo Estable** | no |

|  |  |
| --- | --- |
| **Nombre del Algoritmo** | Quicksort: |
| **Mejor caso** | probabilísticamente garantizado. |
| **Complejidad en el mejor caso (notación O)** | (nlog(n)), |
| **Peor caso** | probabilísticamente garantizado. |
| **Complejidad en el peor caso (notación O)** | O(nlog(n) |
| **Algoritmo Inplace** | si |
| **Algoritmo Adaptativo** | si |
| **Algoritmo Estable** | no |

|  |  |
| --- | --- |
| **Nombre del Algoritmo** | 3-way QuickSort |
| **Mejor caso** | la probabilidad depende de la distribución de las llaves de entrada, pues entre menos objetos sean es mucho más rápido |
| **Complejidad en el mejor caso (notación O)** | O(nlog(n)), |
| **Peor caso** | en este caso se comportaría como un Quicksort normal porque no se aprovecha el tercer camino de igualdad |
| **Complejidad en el peor caso (notación O)** | O(nlogn), |
| **Algoritmo Inplace** | si |
| **Algoritmo Adaptativo** | no |
| **Algoritmo Estable** | no |

|  |  |
| --- | --- |
| **Nombre del Algoritmo** | MergeSort: |
| **Mejor caso** | siempre pasa este caso porque el algoritmo sigue un patrón que no varía mucho, en tanto sigue haciendo las mismas comparaciones. |
| **Complejidad en el mejor caso (notación O)** | O(nlog(n)), |
| **Peor caso** | siempre pasa este caso porque el algoritmo sigue un patrón que no varía mucho, en tanto sigue haciendo las mismas comparaciones. |
| **Complejidad en el peor caso (notación O)** | O(nlog(n)), |
| **Algoritmo Inplace** | no |
| **Algoritmo Adaptativo** | si |
| **Algoritmo Estable** | si |

c.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ShellSort (ms)** | **MergeSort (ms)** | **QuickSort (ms)** |
| **Tiempo Ejecución 1** | 11,384 | 10,263 | 12,474 |
| **Tiempo Ejecución 2** | 11,367 | 10,373 | 10,548 |
| **Tiempo Ejecución 3** | 11,826 | 10,739 | 11,912 |
| **Tiempo Promedio** | 11,525 | 10,458 | 11,644 |

**Conclusión**: Por el tiempo promedio de ejecución, para el caso general, el algoritmo más eficiente es MergeSort. El siguiente algoritmo en eficiencia es ShellSort. El algoritmo menos eficiente es QuickSort.