|  |  |
| --- | --- |
| **Nombre del algoritmo** | Shell Sort |
| **Mejor caso** | El mejor caso se da cuando el arreglo ya está ordenado. |
| **Complejidad en el mejor caso** |  |
| **Peor caso** | El peor caso se da cuando los intervalos en donde se aplica el insertion sort son muy grandes y están totalmente desordenados. |
| **Complejidad en el peor caso** | Depende del caso. |
| **Algoritmo *inplace*** | Sí |
| **Algoritmo adaptativo** | Sí |
| **Algoritmo estable** | No |

|  |  |
| --- | --- |
| **Nombre del algoritmo** | Merge Sort |
| **Mejor caso** | El mejor caso sucede cuando para todo “merge” que se da, el elemento más grande de un subgrupo es más pequeño que el primero del otro subgrupo. |
| **Complejidad en el mejor caso** |  |
| **Peor caso** | El peor caso ocurre cuando se necesitan comparar absolutamente todos los datos de la estructura de datos que se está ordenando. Esto sucede cuando los 2 elementos más grandes de cada grupo son separados en 2 subgrupos diferentes, por lo cual no se salta ninguna comparación. |
| **Complejidad en el peor caso** |  |
| **Algoritmo *inplace*** | No |
| **Algoritmo adaptativo** | No |
| **Algoritmo estable** | Sí |

|  |  |
| --- | --- |
| **Nombre del algoritmo** | Quick Sort |
| **Mejor caso** | El mejor caso se da cuando el pívot queda en la mitad de cada grupo de datos. Adicionalmente, si está casi ordenado la cantidad de cambios es mínima, lo cual optimiza la organización. |
| **Complejidad en el mejor caso** |  |
| **Peor caso** | Cuando el pivote queda en los extremos. También se da cuando existen elementos repetidos. |
| **Complejidad en el peor caso** |  |
| **Algoritmo *inplace*** | Sí |
| **Algoritmo adaptativo** | No |
| **Algoritmo estable** | No |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hora de consulta de viajes | Número de viajes a ordenar | Shellsort (mseg) | Mergesort (mseg) | Quicksort(mseg) |
| 12 | 439,788 | 592 | 312 | 208 |
| 13 | 443,165 | 636 | 300 | 164 |
| 14 | 448,815 | 657 | 308 | 188 |
| 15 | 455,701 | 698 | 327 | 168 |
| 16 | 467,965 | 665 | 320 | 184 |
| 17 | 478,938 | 736 | 348 | 180 |
|  | #Viajes promedio: 455,729 | Tiempo promedio: 664 | Tiempo promedio: 319 | Tiempo promedio:182 |

Por el tiempo promedio de ejecución, para el caso general, el algoritmo más eficiente es Quicksort. El siguiente algoritmo en eficiencia es Mergesort. El algoritmo menos eficiente es Shellsort.