|  |  |
| --- | --- |
| Nombre del Algoritmo | Shell Sort |
| Mejor Caso | Este algoritmo presenta su mejor caso cuando el arreglo esta ordenado o los números que están separados por el espacio más grande después de la primera iteración quedan completamente ordenados.  Depende mucho de la cantidad de espacios, mientras más grande el espacio más afectará. |
| Complejidad Mejor caso | O(nlog(n)) |
| Peor Caso | Este algoritmo presenta su peor caso cuando la mitad alta de los números están situados , de forma ordenada, en las posiciones con índice par y la mitad baja de los números están situados de la misma manera en las posiciones con índice impar. |
| Complejidad Peor caso | ~n^3/2 |
| Algoritmo In Place | Sí |
| Algoritmo Adaptativo | Sí |
| Algoritmo Estable | No |

|  |  |
| --- | --- |
| Nombre del Algoritmo | Merge Sort |
| Mejor Caso | Este algoritmo aunque es usualmente parecido en todos los casos, existe uno en el que se puede contar con suerte. Básicamente es cuando al hacer merge de la mitad derecha del árbol y la mitad izquierda estos ya están ordenados y no hay necesidad de realizar intercambios.  Esto se repite para los subárboles debido a la cantidad menores de intercambio en merge que toca realizar. |
| Complejidad Mejor caso | ~O( nlog(n) / 2 ) |
| Peor Caso | A diferencia del mejor caso aquí no hay una posibilidad específica para que la complejidad empeore, ya que el algoritmo esta hecho para realizar comparaciones muy pequeñas y después a partir de ahí ir ordenando las listas con sus mitades ya ordenadas, por lo que el peor caso en realidad esta cerca de ser el caso promedio. |
| Complejidad Peor caso | ~O( nlog(n) ) |
| Algoritmo In Place | No |
| Algoritmo Adaptativo | No |
| Algoritmo Estable | Sí |

|  |  |
| --- | --- |
| Nombre del Algoritmo | Quick Sort |
| Mejor Caso | Este algoritmo al tener un pivote de por medio para el ordenamiento, depende al final de la suerte y/u optimización con la que se cuente en la elección del pivote.  En el mejor caso, el pivote termina en la mitad de la lista, dividiéndola en dos sublistas de igual tamaño. |
| Complejidad Mejor caso | ~O( nlog(n) ) |
| Peor Caso | En el peor de los casos el pivote termina en un extremo de las listas. Aunque esto usualmente pasa en listas que están ordenadas o casi ordenadas, lo cuál no es extraño ya que la mayoría de optimizaciones del algoritmo están centradas en la elección del pivote, lo que hace que el promedio de complejidad en realidad sea el mejor caso. |
| Complejidad Peor caso | ~O( n^2 / 2 ) |
| Algoritmo In Place | Sí |
| Algoritmo Adaptativo | No |
| Algoritmo Estable | No |